
fhirpath Documentation

Release 0.10.6.dev6+g3b81987

Md Nazrul Islam

Jun 26, 2021

CONTENTS:

1	Introduction	1
1.1	Usages	2
1.2	Available Provider (known)	3
1.3	Elasticsearch Custom Analyzer	4
1.4	ToDo	5
1.5	Credits	5
2	Installation	7
2.1	Stable release	7
2.2	From sources	7
3	Usage	9
3.1	Simple example	9
4	Contributing	11
4.1	Types of Contributions	11
4.2	Get Started!	12
4.3	Pull Request Guidelines	13
4.4	Tips	13
4.5	Deploying	13
5	Credits	15
5.1	Development Lead	15
5.2	Contributors	15
6	History	17
6.1	0.10.6 (unreleased)	17
6.2	0.10.5 (2020-12-17)	17
6.3	0.10.4 (2020-11-19)	17
6.4	0.10.3 (2020-11-17)	17
6.5	0.10.2 (2020-11-06)	18
6.6	0.10.1 (2020-11-04)	18
6.7	0.10.0 (2020-11-04)	18
6.8	0.9.1 (2020-10-24)	18
6.9	0.9.0 (2020-10-24)	18
6.10	0.8.1 (2020-10-05)	19
6.11	0.8.0 (2020-09-25)	19
6.12	0.7.1 (2020-08-07)	19
6.13	0.7.0 (2020-08-07)	19
6.14	0.6.2 (2020-06-30)	20
6.15	0.6.1 (2020-05-09)	20

6.16	0.6.0 (2020-05-08)	20
6.17	0.5.1 (2020-03-18)	20
6.18	0.5.0 (2020-03-11)	20
6.19	0.4.1 (2019-11-05)	21
6.20	0.4.0 (2019-10-24)	21
6.21	0.3.1 (2019-10-08)	21
6.22	0.3.0 (2019-09-30)	22
6.23	0.2.0 (2019-09-15)	22
6.24	0.1.1 (2019-08-15)	22
6.25	0.1.0 (2018-12-15)	22
7	fhirpath	23
7.1	fhirpath package	23
8	Indices and tables	67
	Python Module Index	69
	Index	71

INTRODUCTION



FHIRPath Normative Release (v2.0.0) implementation in Python, along side it provides support for **FHIR Search** API and Query (we called it **fql** (**FHIR Query Language**)) API to fetch FHIR resources from any data-source(database). This library is built in **ORM** like approach. Our goal is to make 100% (as much as possible) **FHIRPath** Normative Release (v2.0.0) specification compliance product.

- Supports FHIR® STU3 and R4.
- Supports multiple provider's engine. Now **Plone** & **guillotina** framework powered providers **fhirpath-guillotina** and **collective.fhirpath** respectively are supported and more coming soon.
- Supports multiple dialects, for example **elasticsearch**, **GraphQL**, **PostgreSQL**. Although now **elasticsearch** has been supported.
- Provide full support of **FHIR Search** with easy to use API.

1.1 Usages

This library is kind of abstract type, where all specifications from [FHIRPath Normative Release \(v2.0.0\)](#) are implemented rather than completed solution (ready to go). The main reason behind this design pattern, to support multiple database systems as well as well as any framework, there is no dependency.

fhirpath never taking care of creating indexes, mappings (elasticsearch) and storing data, if you want to use this library, you have to go through any of existing providers (see list below) or make your own provider (should not too hard work).

1.1.1 Simple example

Assumption:

1. Elasticsearch server 7.x.x Installed.
2. Mappings and indexes are handled manually.
3. Data (document) also are stored manually.

Create Connection and Engine:

```
>>> from fhirpath.connectors import create_connection
>>> from fhirpath.engine.es import ElasticsearchEngine
>>> from fhirpath.engine import dialect_factory
>>> from fhirpath.enums import FHIR_VERSION

>>> host, port = "127.0.0.1", 9200
>>> conn_str = "es://{0}:{1}/".format(host, port)
>>> connection = create_connection(conn_str, "elasticsearch.Elasticsearch")
>>> connection.raw_connection.ping()
True
>>> engine = ElasticsearchEngine(FHIR_VERSION.R4, lambda x: connection, dialect_factory)
```

Basic Search:

```
>>> from fhirpath.search import Search
>>> from fhirpath.search import SearchContext

>>> search_context = SearchContext(engine, "Organization")
>>> params = (
....     ("active", "true"),
....     ("_lastUpdated", "2010-05-28T05:35:56+00:00"),
....     ("_profile", "http://hl7.org/fhir/Organization"),
....     ("identifier", "urn:oid:2.16.528.1|91654"),
....     ("type", "http://hl7.org/fhir/organization-type|prov"),
....     ("address-postalcode", "9100 AA"),
....     ("address", "Den Burg"),
.... )
>>> fhir_search = Search(search_context, params=params)
>>> bundle = fhir_search()
>>> len(bundle.entry) == 0
True
```

Basic Query:

```

>>> from fhirpath.enums import SortOrderType
>>> from fhirpath.query import Q_
>>> from fhirpath.fql import T_
>>> from fhirpath.fql import V_
>>> from fhirpath.fql import exists_
>>> query_builder = Q_(resource="Organization", engine=engine)
>>> query_builder = (
....     query_builder.where(T_("Organization.active") == V_("true"))
....     .where(T_("Organization.meta.lastUpdated", "2010-05-28T05:35:56+00:00"))
....     .sort(sort_("Organization.meta.lastUpdated", SortOrderType.DESC))
.... )
>>> query_result = query_builder(async_result=False)
>>> for resource in query_result:
....     assert resource.__class__.__name__ == "OrganizationModel"
>>> # test fetch all
>>> result = query_result.fetchall()
>>> result.__class__.__name__ == "EngineResult"
True

>>> query_builder = Q_(resource="ChargeItem", engine=engine)
>>> query_builder = query_builder.where(exists_("ChargeItem.enteredDate"))
>>> result = query_builder(async_result=False).single()
>>> result is not None
True
>>> isinstance(result, builder._from[0][1])
True

>>> query_builder = Q_(resource="ChargeItem", engine=engine)
>>> query_builder = query_builder.where(exists_("ChargeItem.enteredDate"))
>>> result = query_builder(async_result=False).first()
>>> result is not None
True
>>> isinstance(result, builder._from[0][1])
True

```

1.2 Available Provider (known)

Currently very few numbers of providers available, however more will coming soon.

1.2.1 fhirpath-guillotina

A guillotina framework powered provider, battery included, ready to go! Please follow associated documentation.

1. **Engine:** Elasticsearch
2. **PyPi:** <https://pypi.org/project/fhirpath-guillotina/>
3. **Source:** https://github.com/nazrulworld/fhirpath_guillotina

1.2.2 collective.fhirpath

A Plone powered provider, like `fhirpath-guillotina` every thing is included. ready to go, although has a dependency on `plone.app.fhirfield`.

1. **Engine:** Elasticsearch
2. **PyPi:** <https://pypi.org/project/collective.fhirpath/>
3. **Source:** <https://github.com/nazrulworld/collective.fhirpath>

1.2.3 unlisted

Why are you waiting for? You are welcome to list your provider here! Developing provider should not be so hard, as `fhirpath` is giving you convenient APIs.

1.3 Elasticsearch Custom Analyzer

To get some special search features for reference type field, you will need to setup custom analyzer for your elasticsearch index.

Example Custom Analyzer:

```
settings = {
  "analysis": {
    "normalizer": {
      "fhir_token_normalizer": {"filter": ["lowercase", "asciifolding"]}
    },
    "analyzer": {
      "fhir_reference_analyzer": {
        "tokenizer": "keyword",
        "filter": ["fhir_reference_filter"],
      },
    },
    "filter": {
      "fhir_reference_filter": {
        "type": "pattern_capture",
        "preserve_original": True,
        "patterns": [r"(?:\w+\/)?(https?:\\.\|\/.*|[a-zA-Z0-9_-]+)"],
      },
    },
    "char_filter": {},
    "tokenizer": {},
  }
}
```

Example Mapping (Reference Field):

```
"properties": {
  "reference": {
    "type": "text",
    "index": true,
    "store": false,
```

(continues on next page)

(continued from previous page)

```
"analyzer": "fhir_reference_analyzer"
}
```

1.4 ToDo

1. [fhirbase](#) engine aka provider implementation.
2. All methods/functions are defined in [FHIRPath](#) specification, would be completed.
3. Implement <https://github.com/ijl/orjson>
4. <https://developers.redhat.com/blog/2017/11/16/speed-python-using-rust/>

1.5 Credits

This package skeleton was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

© Copyright HL7® logo, FHIR® logo and the flaming fire are registered trademarks owned by [Health Level Seven International](#)

“FHIR® is the registered trademark of HL7 and is used with the permission of HL7. Use of the FHIR trademark does not constitute endorsement of this product by HL7” <https://github.com/beda-software/fhirpath-py>

INSTALLATION

2.1 Stable release

To install fhirpath, run this command in your terminal:

```
$ pip install fhirpath
```

This is the preferred method to install fhirpath, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for fhirpath can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/nazrulworld/fhirpath
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/nazrulworld/fhirpath/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```


USAGE

This library is kind of abstract type, where all specifications from `fhirpath` are implemented rather than completed solution (ready to go). The main reason behind this design pattern, to support multiple database systems as well as well as any framework, there is no dependency.

`fhirpath` never taking care of creating indexes, mappings (elasticsearch) and storing data, if you want to use this library, you have to go through any of existing providers (see list bellow) or make your own provider (should not too hard work).

3.1 Simple example

Assumption:

1. Elasticsearch server 7.x.x Installed.
2. Mappings and indexes are handled manually.
3. Data (document) also are stored manually.

Create Connection and Engine:

```
>>> from fhirpath.connectors import create_connection
>>> from fhirpath.engine.es import ElasticsearchEngine
>>> from fhirpath.engine import dialect_factory
>>> from fhirpath.enums import FHIR_VERSION

>>> host, port = "127.0.0.1", 9200
>>> conn_str = "es://{0}:{1}/".format(host, port)
>>> connection = create_connection(conn_str, "elasticsearch.Elasticsearch")
>>> connection.raw_connection.ping()
True
>>> engine = ElasticsearchEngine(FHIR_VERSION.R4, lambda x: connection, dialect_factory)
```

Basic Search:

```
>>> from fhirpath.search import Search
>>> from fhirpath.search import SearchContext

>>> search_context = SearchContext(engine, "Organization")
>>> params = (
....     ("active", "true"),
....     ("_lastUpdated", "2010-05-28T05:35:56+00:00"),
```

(continues on next page)

(continued from previous page)

```

....     ("_profile", "http://hl7.org/fhir/Organization"),
....     ("identifier", "urn:oid:2.16.528.1|91654"),
....     ("type", "http://hl7.org/fhir/organization-type|prov"),
....     ("address-postalcode", "9100 AA"),
....     ("address", "Den Burg"),
.... )
>>> fhir_search = Search(search_context, params=params)
>>> bundle = fhir_search()
>>> len(bundle.entry) == 0
True

```

Basic Query:

```

>>> from fhirpath.enums import SortOrderType
>>> from fhirpath.query import Q_
>>> from fhirpath.fql import T_
>>> from fhirpath.fql import V_
>>> from fhirpath.fql import exists_
>>> query_builder = Q_(resource="Organization", engine=engine)
>>> query_builder = (
....     query_builder.where(T_("Organization.active") == V_("true"))
....     .where(T_("Organization.meta.lastUpdated", "2010-05-28T05:35:56+00:00"))
....     .sort(sort_("Organization.meta.lastUpdated", SortOrderType.DESC))
.... )
>>> query_result = query_builder(async_result=False)
>>> for resource in query_result:
....     assert resource.__class__.__name__ == "OrganizationModel"
>>> # test fetch all
>>> result = query_result.fetchall()
>>> result.__class__.__name__ == "EngineResult"
True

>>> query_builder = Q_(resource="ChargeItem", engine=engine)
>>> query_builder = query_builder.where(exists_("ChargeItem.enteredDate"))
>>> result = query_builder(async_result=False).single()
>>> result is not None
True
>>> isinstance(result, builder._from[0][1])
True

>>> query_builder = Q_(resource="ChargeItem", engine=engine)
>>> query_builder = query_builder.where(exists_("ChargeItem.enteredDate"))
>>> result = query_builder(async_result=False).first()
>>> result is not None
True
>>> isinstance(result, builder._from[0][1])
True

```

CONTRIBUTING

`fhirpath` specification is really big job to be done completely, quite impossible to done for single person effort. This library currently is in very early stage! but have solid starting point. Contributions are needed and welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at <https://github.com/nazrulworld/fhirpath/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

4.1.4 Write Documentation

fhirpath could always use more documentation, whether as part of the official fhirpath docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/nazrulworld/fhirpath/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *fhirpath* for local development.

1. Fork the *fhirpath* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/fhirpath.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ cd fhirpath/  
$ pipenv install --dev --pre -e .[test]  
$ pipenv shell  
$ buildout -c buildout.cfg
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 fhirpath tests  
$ python setup.py test or py.test  
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .  
$ git commit -m "Your detailed description of your changes."  
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/nazrulworld/fhirpath/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CREDITS

5.1 Development Lead

- Md Nazrul Islam <email2nazrul@gmail.com> (Author)

5.2 Contributors

- Jason Paumier <jason@arkhn.com>
- Simon Vadée <simon@arkhn.com>

HISTORY

6.1 0.10.6 (unreleased)

- Switch to Apache Software License v2

6.2 0.10.5 (2020-12-17)

Improvement

- `BundleWrapper` is providing two optional methods, `calculate_fhir_base_url` and `resolve_absolute_uri` and is now also taking optional parameter `base_url`.

Fixes - fixes `ElasticSearchDialect.create_term` [Kartik Sayani]

- fixes `EngineResult.extract_references`. [Jason Paumier]
- fixes how composite search params are parsed. [Jason Paumier]
- Issue#28 [Nested GroupTerm search returns no matches](#)
- fixes `SearchContext.normalize_param` for composite search params [nazrulworld]

6.3 0.10.4 (2020-11-19)

- fixes `FHIRAbstractModel` comparing at `utils` module for `BundleWrapper`.
- `fallback_callable` helper function is available at `utils` module.

6.4 0.10.3 (2020-11-17)

Improvements

- More helper functions (`get_local_timezone`, `timestamp_utc`, `timestamp_utc`) are created.
- `initial_bundle_data` method is now available in Base Elasticsearch engine class, meaning that it is possible construct Bundle initial data into the derived class, so more flexibility.

Bugfixes

- Default bundle initial data was constructed `meta.lastUpdated` value with `utc` now time but without timezone offset, so during json serialization, timezone info was missed as a result reverse construct of Bundle complains validation error.

6.5 0.10.2 (2020-11-06)

Improvements

- `orjson` is no longer required. `json_dumps` and `json_loads` now dynamically supports `orjson` and `simplejson`.

6.6 0.10.1 (2020-11-04)

Bugfixes

- `Connection.raw_connection` was wrongly wrapped by `AsyncElasticsearchConnection/ElasticsearchConnection.from_url()` with `self`, instead of `elasticsearch.AsyncElasticsearch/elasticsearch.Elasticsearch`'s instance.

6.7 0.10.0 (2020-11-04)

Improvements

- Introducing `AsyncElasticsearchConnection` and `AsyncElasticsearchEngine` the `asyncio` based connection and engine for Elasticsearch. See [Using Asyncio with Elasticsearch](#)
- Added `orjson` based json serializer for Elasticsearch by default (when connection is made from connection factory).
- Added support for `_summary=text|data|count|true|false`. [arkhn]
- Added support for `_elements` search parameter. [arkhn]

Breaking

- `async_result` parameter is no longer needed for `SearchContext`, `Search` and `Query` (included `async` version) as from now all engine contains that information (`engine_class.is_async()`).

6.8 0.9.1 (2020-10-24)

- Added supports for `_format` and `_pretty` params, now there should no complain for those, instead of simply ignored. [nazrulworld]

6.9 0.9.0 (2020-10-24)

- Handle `:identifier` modifier for reference search parameters [simonvadee]
- fixes `BundleWrapper` as `_json` mode, now includes with `resourceType` value. [nazrulworld]
- Dict response option has been added in `fhirpath.search.fhir_search` [nazrulworld]
- Ignore empty search params #21 [simonvadee]
- Just for performance optimization issue minimum required `zope.interface` version is 5.1.2.

6.10 0.8.1 (2020-10-05)

- Disable pydantic validation for Bundle in `fhirpath.utils.BundleWrapper` [simonvadee]
- Two helper functions `json_dumps` and `json_loads` are now available under `utils` module [nazrulworld]
- Only apply search prefix on affected types #17 [simonvadee]

6.11 0.8.0 (2020-09-25)

Improvements

- add supports for some important FHIR search parameters (`_has`, `_include` and `_revinclude`) [simonvadee]
- enable search on several resource types (`_type` search param) [Jasopaum]
- Issue #8 Add search support for without any params or query string if context has resource type [nazrulworld]
- Issue #9 multiple negative not working [nazrulworld]

Breaking

- `fhirpath.search.SearchContext.resource_name` has been changed `fhirpath.search.SearchContext.resource_type` and now datatype is List instead of string. Please check your API. [Jasopaum]
- For case of Elasticsearch based engine, you should use custom analyzer (`fhir_reference_analyzer`) for FHIR Reference type. For details see readme.

6.12 0.7.1 (2020-08-07)

- added missing `isodate` package dependency.

6.13 0.7.0 (2020-08-07)

Improvements

- Issue#5: Now `ElasticsearchEngine::get_index_name` takes one optional parameter `resource_type`.
- Add supports for python version 3.6.

Breaking

- Make full capability with `fhir.resources` version 6.x.x, please have a look of revolutionary changes of `fhir.resources`.

6.14 0.6.2 (2020-06-30)

- `fhirspec` and `fhir.resources` versions are pinned.

6.15 0.6.1 (2020-05-09)

A must update release (from 0.6.0)!

Bugfixes

- fix: issues, those arises due to fix below issue.
- fix: `fhirpath.storage.FHIR_RESOURCE_CLASS_STORAGE`, `fhirpath.storage.PATH_INFO_STORAGE`, `fhirpath.storage.SEARCH_PARAMETERS_STORAGE` and `fhirpath.storage.FHIR_RESOURCE_SPEC_STORAGE` took wrong FHIR release as keys.

6.16 0.6.0 (2020-05-08)

Breaking

- Hard dependency on `fhirspec`.
- Minimum python version 3.7 is required.
- Minimum required `fhir.resources` version is now 5.1.0 meaning FHIR R4 4.0.1 and STU3 3.0.2. Please follow changes log <https://pypi.org/project/fhir.resources/5.1.0/>.

6.17 0.5.1 (2020-03-18)

New features

- `__main__` module has been created, now possible to see version and/or initiated required FHIR versions. For example `python -m "fhirpath" --version`, `python -m "fhirpath" --init-setup [nazrulworld]`

Improvements

- Updated fix version of elasticsearch mappings.

6.18 0.5.0 (2020-03-11)

New Features

- `FHIRPath` (Normative Release) support available. A dedicated class is now available `fhirpath.FHIRPath`, although it is working in progress (meaning that many methods/functions are yet to do complete.)

Improvements

- Add support for important FHIR search modifier `:contains`. See <https://github.com/nazrulworld/fhirpath/issues/1>
- Add support for `:above` FHIR search modifier and ``eb` prefix. See <https://github.com/nazrulworld/fhirpath/issues/2>

- Add support for :bellow FHIR search modifier and sa prefix. See <https://github.com/nazrulworld/fhirpath/issues/2>

Bugfixes

- Upgrade to this version is recommended as it includes couples of major bug fixes.

Breaking

- The `fhirpath.navigator` module has been removed and introduced new module `fhirpath.model`. `fhirpath.utils.Model` has been moved to `fhirpath.model``.

6.19 0.4.1 (2019-11-05)

Bugfixes

- `fhirpath.search.Search.parse_query_string` now returning `MuliDict```(what is expected) instead of ```MultiDictProxy`.

6.20 0.4.0 (2019-10-24)

Improvements

- Now full select features are accepted, meaning that you can provide multiple path in select section. for example `select(Patient.name, Patient.gender)`.
- `FHIRPath.count()` and `empty()` functions are supported.
- Supports path navigation with index and functions inside select. Example `[index]`, `last()`, `first()`, `Skip()`, `Take()`, `count()`.

Breakings

- `QueryResult.first` and `QueryResult.single` are no longer return FHIR Model instance instead returning `fhirpath.engine.EngineResultRow`.
- `QueryResult.fetchall` returning list of `fhirpath.engine.EngineResultRow` instead of FHIR JSON.
- `QueryResult` iteration returning list of FHIR Model instance on condition (if select is `*`), other than returning list of `fhirpath.engine.EngineResultRow`.

6.21 0.3.1 (2019-10-08)

Improvements

- Add support for search parameter expression that contains with space+as (`MedicationRequest.medication as CodeableConcept`)

Bugfixes

- `not` modifier is now working for `Coding` and `CodeableConcept`.
- `“ignore_unmapped”` now always `True` in case of nested query.
- `“unmapped_type”` now set explicitly long value. See related issue <https://stackoverflow.com/questions/17051709/no-mapping-found-for-field-in-order-to-sort-on-in-elasticsearch>

6.22 0.3.0 (2019-09-30)

Improvements

- Supports multiple AND values for same search parameter!.
- Add support FHIR version STU3 compability for Money type search.[nazrulworld]
- IN Query support added.[nazrulworld]
- Support PathElement that contains string path with .as(), thus supports for Search also.
- Supports Duration type in Search.
- Add support composite type search param.

Bugfixes

- Multiple search values (IN search)
- Missing text for HumanName and Address search.

6.23 0.2.0 (2019-09-15)

Breakings:

- Built-in providers (`guillotina_app` and `plone_app`) have been wiped as both becoming separate pypi project.
- `queries` module has been moved from `fql` sub-package to `fhirpath` package and also renamed as `query`.

Improvements:

- There are so many improvements made for almost all most modules.
- FhirSearch coverages are increased.
- Sort, Limit facilities added in Query as well in FhirSearch.

Bugfixes:

- numbers of bugs fixed.

6.24 0.1.1 (2019-08-15)

- First working version has been released. Of-course not full featured.

6.25 0.1.0 (2018-12-15)

- First release on PyPI.(Just register purpose, not usable at all, next release coming soon)

FHIRPATH

7.1 fhirpath package

7.1.1 Subpackages

fhirpath.connectors package

Subpackages

fhirpath.connectors.factory package

Submodules

fhirpath.connectors.factory.es module

fhirpath.connectors.factory.pg module

Module contents

```
class fhirpath.connectors.factory.ConnectionFactory(url, klass, **extra)
    Bases: object
        wrap(raw_conn)
```

Submodules

fhirpath.connectors.connection module

```
class fhirpath.connectors.connection.Connection(conn)
    Bases: object
        classmethod from_config(config: dict)
        classmethod from_prepared(conn)
            Connection instance creation, using already prepared RAW connection
        classmethod from_url(url: str)
            1.) may be use connector utilities 2.) may be url parser
        classmethod is_async()
```

property `raw_connection`

fhirpath.connectors.interfaces module

fhirpath.connectors.url module

Idea has been take from SQLAlchemy @from: sqlalchemy/engine/url.py But we modified according to our requirements. _____ Copyright (C) 2005-2019 the SQLAlchemy authors and contributors <see AUTHORS file> This module is part of SQLAlchemy and is released under the MIT License: <http://www.opensource.org/licenses/mit-license.php> _____

class fhirpath.connectors.url.URL(*drivername, username=None, password=None, host=None, port=None, database=None, query=None*)

Bases: object

Represent the components of a URL used to connect to a database.

This object is suitable to be passed directly to a `create_engine()` call. The fields of the URL are parsed from a string by the `make_url()` function. the string format of the URL is an RFC-1738-style string.

All initialization parameters are available as public attributes.

Parameters

- **drivername** – the name of the database backend. This name will correspond to a module in sqlalchemy/databases or a third party plug-in.
- **username** – The user name.
- **password** – database password.
- **host** – The name of the host.
- **port** – The port number.
- **database** – The database name.
- **query** – A dictionary of options to be passed to the dialect and/or the DBAPI upon connect.

`get_backend_name()`

`get_driver_name()`

property `password`

`translate_connect_args(names=None, **kw)`

Translate url attributes into a dictionary of connection arguments.

Returns attributes of this url (*host, database, username, password, port*) as a plain dictionary. The attribute names are used as the keys by default. Unset or false attributes are omitted from the final dictionary.

Parameters

- ****kw** – Optional, alternate key names for url attributes.
- **names** – Deprecated. Same purpose as the keyword-based alternate names, but correlates the name to the original positionally.

fhirpath.connectors.url.to_list(*x, default=None*)

Module contents

`fhirpath.connectors.create_connection(conn_string, klass=None, **extra)`

`fhirpath.connectors.make_url(connection_str)`

fhirpath.dialects package

Submodules

fhirpath.dialects.base module

`class fhirpath.dialects.base.DialectBase(connection=None)`

Bases: `object`

`bind(connection)`

`compile(query, mapping=None, root_replacer=None, **kwargs)`

`static is_fhir_primitive_type(klass)`

`pre_compile(query)`

xxx: validation placeholder

fhirpath.dialects.elasticsearch module

ElasticSearch Dialect

`class fhirpath.dialects.elasticsearch.ElasticSearchDialect(connection=None)`

Bases: `fhirpath.dialects.base.DialectBase`

`static apply_from_constraint(query, body_structure, resource_type, root_replacer=None)`

We force apply resource type boundary

`static apply_limit(limit_clause, body_structure)`

`static apply_nested(query, dotted_path)`

`static apply_path_replacement(dotted_path, root_replacer)`

`static apply_sort(sort_terms, body_structure, root_replacer=None)`

`static apply_source_filter(query, body_structure, root_replacer=None)`

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html#request-body-search-source-filtering>

- 1.) **we are using FHIR field data from ES server directly, unlike collective.** elasticsearch, where only path is retrieve, then using that set zcatalog brain, this patternt might good for general puporse but here we exclusively need fhir resource only which is already stored in ES. Our approach will be defnately performance optimized!

- 2.) **We might loose minor security (only zope specific),** because here permission is not checking while getting full object.

`static attach_nested_on_demand(context, query_, root_replacer)`

`static clean_up(body_structure)`

`compile(query, calculate_field_index_name, get_mapping)`

compile_for_single_resource_type(*query*, *resource_type*, *mapping=None*, *root_replacer=None*)

Param *query*

Param *mapping*: Elasticsearch mapping for FHIR resources.

Root_replacer Path's root replacer: Could be mapping name or index name in zope's ZCatalog context

static create_contains_term(*path*, *value*)

Create ES Regex Query

static create_dotted_path(*term*, *root_replacer=None*)

static create_eb_term(*path*, *value*)

Create ES Prefix Query

static create_sa_term(*path*, *value*)

Create ES Prefix Query

static create_structure()

static create_term(*path*, *value*, *multiple=False*, *match_type=None*, *all_resources=False*)

Create ES Query term

static get_path_mapping_info(*mapping*, *dotted_path*)

resolve_datetime_term(*term*, *root_replacer=None*)

static resolve_exists_term(*term*, *root_replacer=None*)

resolve_nonfhir_term(*term*)

static resolve_numeric_term(*term*, *root_replacer=None*)

static resolve_string_term(*term*, *map_info*, *root_replacer=None*)

resolve_term(*term*, *mapping*, *root_replacer*)

`fhirpath.dialects.elasticsearch.escape_all(v)`

`fhirpath.dialects.elasticsearch.escape_star(v)`

fhirpath.dialects.postgres module

RAW PostgreSQL Dialect for FHIRPath Engine

class `fhirpath.dialects.postgres.PostgresDialect`(*connection=None*)

Bases: `fhirpath.dialects.base.DialectBase`

fhirpath.dialects.sqlalchemy module

class `fhirpath.dialects.sqlalchemy.SqlAlchemyDialect`(*connection=None*)

Bases: `fhirpath.dialects.base.DialectBase`

Module contents

fhirpath.engine package

Submodules

fhirpath.engine.base module

```
class fhirpath.engine.base.Engine(fhir_release, conn_factory, dialect_factory)
    Bases: abc.ABC

    Idea: # 1.) https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.Engine.connect 2.)
    https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.Connection 3.) Dialect could have
    raw connection, query compiler 4.) Engine would have execute and result processing through provider, yes
    provider!

    before_execute(query)
        Hook: before execution of query

    create_connection(factory)

    create_dialect(factory)

    classmethod is_async()

class fhirpath.engine.base.EngineResult(header: fhirpath.engine.base.EngineResultHeader, body:
                                         fhirpath.engine.base.EngineResultBody)
    Bases: object

    body: fhirpath.engine.base.EngineResultBody

    extract_ids() → Dict[str, List[str]]

    extract_references(search_param: fhirpath.fhirspec.spec.SearchParameter) → Dict[str, List[str]]
        Takes a search parameter as input and extract all targeted references

        Returns a dict like: {"Patient": ["list", "of", "referenced", "patient", "ids"], "Observation": []}

    header: fhirpath.engine.base.EngineResultHeader

class fhirpath.engine.base.EngineResultBody
    Bases: collections.deque

    add(value)

    append(value)

class fhirpath.engine.base.EngineResultHeader(total, raw_query=None)
    Bases: object

    elements = None

    generated_on = None

    raw_query = None

    total = None

class fhirpath.engine.base.EngineResultRow(iterable=(), /)
    Bases: list
```

fhirpath.engine.es module

```
class fhirpath.engine.es.AsyncElasticsearchEngine(fhir_release, conn_factory, dialect_factory)
    Bases: fhirpath.engine.es.ElasticsearchEngineBase
    Async Elasticsearch Engine
    async execute(query, unrestricted=False, query_type=<EngineQueryType.DML: 'DML'>)
    classmethod is_async()
    async process_raw_result(rawresult, selects, query_type)

class fhirpath.engine.es.ElasticsearchEngine(fhir_release, conn_factory, dialect_factory)
    Bases: fhirpath.engine.es.ElasticsearchEngineBase
    Elasticsearch Engine
    execute(query, unrestricted=False, query_type=<EngineQueryType.DML: 'DML'>)
    process_raw_result(rawresult, selects, query_type)

class fhirpath.engine.es.ElasticsearchEngineBase(fhir_release, conn_factory, dialect_factory)
    Bases: fhirpath.engine.base.Engine
    build_security_query(query)
    calculate_field_index_name(resource_type)
    current_url()
        complete url from current request return yarl.URL
    extract_hits(source_filters, hits, container, doc_type='_doc')
    generate_mappings(reference_analyzer: Optional[str] = None, token_normalizer: Optional[str] = None)
        You may use this function to build the ES mapping. Returns an object like: {
            "Patient": {
                "properties": {
                    "identifier": {
                        "properties": {
                            "use": { "type": "keyword", "index": true, "store": false, "fields": {
                                "raw": { "type": "keyword"
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    get_index_name(resource_type: Optional[str] = None)
    get_mapping(resource_type)
```

initial_bundle_data()

Can be overridden in sub class

wrapped_with_bundle(*result*, *includes=None*, *as_json=False*)

`fhirpath.engine.es.navigate_indexed_path(source, path_)`

fhirpath.engine.fhirbase module

class `fhirpath.engine.fhirbase.FhirbaseEngine`(*fhir_release*, *conn_factory*, *dialect_factory*)

Bases: `fhirpath.engine.base.Engine`

Module contents

Implementation, the medium result collected from ES server

class `fhirpath.engine.Connection`(*conn*)

Bases: `object`

classmethod `from_config`(*config: dict*)

classmethod `from_prepared`(*conn*)

Connection instance creation, using already prepared RAW connection

classmethod `from_url`(*url: str*)

1.) may be use connector utilities 2.) may be url parser

classmethod `is_async`()

property `raw_connection`

class `fhirpath.engine.Engine`(*fhir_release*, *conn_factory*, *dialect_factory*)

Bases: `abc.ABC`

Idea: # 1.) <https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.Engine.connect> 2.) <https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.Connection> 3.) Dialect could have raw connection, query compiler 4.) Engine would have execute and result processing through provider, yes provider!

before_execute(*query*)

Hook: before execution of query

create_connection(*factory*)

create_dialect(*factory*)

classmethod `is_async`()

class `fhirpath.engine.EngineResult`(*header: fhirpath.engine.base.EngineResultHeader*, *body: fhirpath.engine.base.EngineResultBody*)

Bases: `object`

body: `fhirpath.engine.base.EngineResultBody`

extract_ids() → `Dict[str, List[str]]`

extract_references(*search_param: fhirpath.fhirspec.spec.SearchParameter*) → `Dict[str, List[str]]`

Takes a search parameter as input and extract all targeted references

Returns a dict like: {"Patient": ["list", "of", "referenced", "patient", "ids"], "Observation": []}

header: `fhirpath.engine.base.EngineResultHeader`

```
class fhirpath.engine.EngineResultBody
    Bases: collections.deque
    add(value)
    append(value)
class fhirpath.engine.EngineResultHeader(total, raw_query=None)
    Bases: object
    elements = None
    generated_on = None
    raw_query = None
    total = None
class fhirpath.engine.EngineResultRow(iterable=(), /)
    Bases: list
```

fhirpath.fhirspec package

Submodules

fhirpath.fhirspec.downloader module

```
fhirpath.fhirspec.downloader.download_and_extract(release: fhirpath.enums.FHIR_VERSION,
                                                    output_dir: pathlib.Path)
fhirpath.fhirspec.downloader.download_archive(release: fhirpath.enums.FHIR_VERSION,
                                                temp_location: pathlib.Path) → pathlib.Path
fhirpath.fhirspec.downloader.extract_spec_files(extract_location: pathlib.Path, archive_file:
                                                  pathlib.Path)
```

fhirpath.fhirspec.spec module

Most of codes are copied from <https://github.com/nazrulworld/fhir-parser> and modified in terms of styling, unnecessary codes cleanup (those are not relevant for this package)

```
class fhirpath.fhirspec.spec.FHIRSearchSpec(source: pathlib.Path, fhir_release:
                                              fhirpath.enums.FHIR_VERSION, storage:
                                              fhirpath.storage.MemoryStorage)
    Bases: object
    https://www.hl7.org/fhir/searchparameter-registry.html
    apply_base_resource_params()
    property jsonfilename
    prepare()
    write()
class fhirpath.fhirspec.spec.ResourceSearchParameterDefinition(resource_type)
    Bases: object
    resource_type
```

```
class fhirpath.fhirspec.spec.SearchParameter
    Bases: object
    clone()
    code
    comparator
    component
    expression
    classmethod from_definition(resource_type, definition)
    get_expression(resource_type, definition)
    modifier
    multiple_and
    multiple_or
    name
    support_prefix()
    target
    type
    xpath

class fhirpath.fhirspec.spec.SearchParameterDefinition
    Bases: object
    code
    comparator
    component
    expression_map
    classmethod from_dict(spec, dict_value)
    modifier
    multiple_and
    multiple_or
    name
    spec
    target
    type
    xpath
```

Module contents

FHIR Specification: <http://www.hl7.org/fhir/>

class fhirpath.fhirspec.FHIRSearchSpecFactory

Bases: object

static from_release(release: str)

class fhirpath.fhirspec.FhirSpecFactory

Bases: object

static from_release(release: str, config: Optional[fhirspec.Configuration] = None)

fhirpath.fhirspec.ensure_spec_jsons(release: fhirpath.enums.FHIR_VERSION)

fhirpath.fhirspec.lookup_fhir_resource_spec(resource_type: str, cache: bool = True, fhir_release: fhirpath.enums.FHIR_VERSION = <FHIR_VERSION.DEFAULT: 'R4'>) → Optional[fhirspec.FHIRStructureDefinition]

Parameters

- **resource_type** – the resource type name (required). i.e Organization
- **cache** – (default True) the flag which indicates should query fresh or serve from cache if available.
- **fhir_release** – FHIR Release (version) name. i.e FHIR_VERSION.STU3, FHIR_VERSION.R4

:return FHIRStructureDefinition

Example:

```
>>> from fhirpath.fhirspec import lookup_fhir_resource_spec
>>> from zope.interface import Invalid
>>> dotted_path = lookup_fhir_resource_spec('Patient')
>>> 'fhir.resources.patient.Patient' == dotted_path
True
>>> dotted_path = lookup_fhir_resource_spec('FakeResource')
>>> dotted_path is None
True
```

fhirpath.fql package

Submodules

fhirpath.fql.expressions module

fhirpath.fql.expressions.G(*terms, path=None, type_=None)

fhirpath.fql.expressions.T(path, value=<NO_VALUE>, match_type=None, non_fhir=False)

fhirpath.fql.expressions.V(value)

fhirpath.fql.expressions.and_(path, value=<NO_VALUE>)

fhirpath.fql.expressions.exists_(path)

fhirpath.fql.expressions.exists_group(*terms, type_=None)

```
fhirpath.fql.expressions.fql(obj)
fhirpath.fql.expressions.in_(path, values)
fhirpath.fql.expressions.not_(path, value=<NO_VALUE>)
fhirpath.fql.expressions.not_exists_(path)
fhirpath.fql.expressions.or_(path, value=<NO_VALUE>)
fhirpath.fql.expressions.sort_(path, order=<NO_VALUE>)
fhirpath.fql.expressions.xor_(path, value=<NO_VALUE>)
```

fhirpath.fql.types module

```
class fhirpath.fql.types.BaseTerm(path, value=<NO_VALUE>, match_type=None)
    Bases: abc.ABC
    clone()
    ensure_term_value(value)
    finalize(context)
    get_real_value()
    set_match_type(type_)
    validate()

class fhirpath.fql.types.ElementClause
    Bases: fhirpath.fql.types.FqlClause

class fhirpath.fql.types.ElementPath(dotted_path: str, non_fhir: bool = False)
    Bases: object
    FHIR Resource path (dotted) 1. Normalize any condition, casting, logic check
    clone()
    finalize(context)
    classmethod from_el_path(el_path)
    is_finalized()
    property non_fhir
    parse()
    property path
    property star
    validate(fhir_release)

class fhirpath.fql.types.ExistsGroupTerm(*terms)
    Bases: object
    clone()
    finalize(context)
    match_all()
    match_any()
```

```
    match_no_one()
    match_one()
class fhirpath.fql.types.ExistsTerm(path)
    Bases: object
    clone()
    finalize(context)
class fhirpath.fql.types.FqlClause
    Bases: collections.deque
    property empty
class fhirpath.fql.types.FromClause
    Bases: fhirpath.fql.types.FqlClause
class fhirpath.fql.types.GroupTerm(*terms, path=None)
    Bases: object
    clone()
    finalize(context)
    match_all()
    match_any()
    match_no_one()
    match_one()
class fhirpath.fql.types.InTerm(path, value=<NO_VALUE>)
    Bases: fhirpath.fql.types.Term
    The InTerm never influences by TermValue unary_operator!
    create_term(value)
    “
    finalize(context)
class fhirpath.fql.types.LimitClause
    Bases: abc.ABC
    property empty
    property limit
    property offset
class fhirpath.fql.types.NonFhirTerm(path, value=<NO_VALUE>, match_type=None)
    Bases: fhirpath.fql.types.BaseTerm
    ensure_term_value(value)
    finalize(context)
    get_real_value()
    validate()
class fhirpath.fql.types.PathWhereConstraint(type_, name=None, value=None, subpath=None)
    Bases: object
    classmethod from_expression(expression)
```

```
class fhirpath.fql.types.SelectClause
    Bases: fhirpath.fql.types.FqlClause

class fhirpath.fql.types.SortClause
    Bases: fhirpath.fql.types.FqlClause

class fhirpath.fql.types.SortTerm(path, order=<SortOrderType.ASC: 'asc'>)
    Bases: object

    finalize(context)

    order = None

    path = None

class fhirpath.fql.types.Term(path, value=<NO_VALUE>, match_type=None)
    Bases: fhirpath.fql.types.BaseTerm

    ensure_term_value(value)

    finalize(context)

    get_real_value()

    validate()

class fhirpath.fql.types.TermValue(value)
    Bases: object

    clone()

    finalize(path)
        context: PathInfoContext

    is_finalized()

class fhirpath.fql.types.WhereClause
    Bases: fhirpath.fql.types.FqlClause
```

Module contents

FHIR Query Language

```
class fhirpath.fql.ElementPath(dotted_path: str, non_fhir: bool = False)
    Bases: object

    FHIR Resource path (dotted) 1. Normalize any condition, casting, logic check

    clone()

    finalize(context)

    classmethod from_el_path(el_path)

    is_finalized()

    property non_fhir

    parse()

    property path

    property star

    validate(fhir_release)
```

```
fhirpath.fql.G_(*terms, path=None, type_=None)
fhirpath.fql.T_(path, value=<NO_VALUE>, match_type=None, non_fhir=False)
fhirpath.fql.V_(value)
fhirpath.fql.and_(path, value=<NO_VALUE>)
fhirpath.fql.contains_(path, value=<NO_VALUE>)
fhirpath.fql.eb_(path, value=<NO_VALUE>)
fhirpath.fql.exact_(path, value=<NO_VALUE>)
fhirpath.fql.exists_(path)
fhirpath.fql.in_(path, values)
fhirpath.fql.not_(path, value=<NO_VALUE>)
fhirpath.fql.not_exists_(path)
fhirpath.fql.not_in_(path, values)
fhirpath.fql.or_(path, value=<NO_VALUE>)
fhirpath.fql.sa_(path, value=<NO_VALUE>)
fhirpath.fql.sort_(path, order=<NO_VALUE>)
```

fhirpath.interfaces package

Submodules

fhirpath.interfaces.base module

fhirpath.interfaces.connectors module

fhirpath.interfaces.dialects module

fhirpath.interfaces.engine module

fhirpath.interfaces.fql module

Module contents

fhirpath.thirdparty package

Submodules

fhirpath.thirdparty.peewee module

Copied from [peewee](#) with little customization

```
class fhirpath.thirdparty.peewee.Proxy
```

Bases: `object`

Create a proxy or placeholder for another object.

attach_callback(*callback*)
initialize(*obj*)
obj
passthrough()

fhirpath.thirdparty.werkzeug module

https://github.com/pallets/werkzeug/blob/master/src/werkzeug/_compat.py

class fhirpath.thirdparty.werkzeug.ImmutableDict

Bases: [fhirpath.thirdparty.werkzeug.ImmutableDictMixin](#), dict

An immutable dict.

New in version 0.5.

copy()

Return a shallow mutable copy of this object. Keep in mind that the standard library's [copy\(\)](#) function is a no-op for this class like for any other python immutable type (eg: tuple).

class fhirpath.thirdparty.werkzeug.ImmutableDictMixin

Bases: object

Makes a dict immutable.

New in version 0.5.

Private

clear()

classmethod **fromkeys**(*keys*, *value=None*)

pop(*key*, *default=None*)

popitem()

setdefault(*key*, *default=None*)

update(**args*, ***kwargs*)

fhirpath.thirdparty.werkzeug.**is_immutable**(*self*)

fhirpath.thirdparty.werkzeug.**iteritems**(*d*, **args*, ***kwargs*)

Module contents

Basically here things are copied from various open source projects

7.1.2 Submodules

7.1.3 fhirpath.cli module

7.1.4 fhirpath.constraints module

fhirpath.constraints.required_finalized(obj: object) → None

fhirpath.constraints.required_from_resource(obj: object) → None

fhirpath.constraints.required_not_finalized(obj: object) → None

fhirpath.constraints.required_value_not_assigned(obj: object) → None

7.1.5 fhirpath.enums module

```
class fhirpath.enums.EngineQueryType(value)
```

Bases: enum.Enum

“

COUNT: str = 'COUNT'

DDL: str = 'DDL'

DML: str = 'DML'

```
class fhirpath.enums.FHIR_VERSION(value)
```

Bases: enum.Enum

Release:str: Version:str

DEFAULT: str = 'R4'

DSTU2: str = '1.0.2'

R4: str = '4.0.1'

STU3: str = '3.0.2'

static normalize(item)

```
class fhirpath.enums.GroupType(value)
```

Bases: enum.Enum

An enumeration.

COUPLED: str = 'COUPLED'

DECOUPLED: str = 'DECOUPLED'

```
class fhirpath.enums.MatchType(value)
```

Bases: enum.Enum

ALL: str = 'ALL'

ANY: str = 'ANY'

NONE: str = 'NONE'

ONE: str = 'ONE'

```
class fhirpath.enums.OPERATOR(value)
```

Bases: enum.Enum

and_: Callable = <built-in function and_>

ap(*b: Any*) → Any

approximately the value for the parameter in the resource is approximately the same to the provided value.
Note that the recommended value for the approximation is 10% of the stated value (or for a date, 10% of the gap between now and the date), but systems may choose other values where appropriate

concat: Callable = <built-in function concat>

contains: Callable = <built-in function contains>

eb(*b: Any*) → Any

ends-before the value for the parameter in the resource ends before the provided value the range of the search value does overlap not with the range of the target value, and the range below the search value contains the range of the target value

eq: Callable = <built-in function eq>

ge: Callable = <built-in function ge>

gt: Callable = <built-in function gt>

le: Callable = <built-in function le>

lt: Callable = <built-in function lt>

ne: Callable = <built-in function ne>

neg: Callable = <built-in function neg>

not_: Callable = <built-in function not_>

or_: Callable = <built-in function or_>

pos: Callable = <built-in function pos>

sa(*b: Any*) → Any

starts-after the value for the parameter in the resource starts after the provided value the range of the search value does not overlap with the range of the target value, and the range above the search value contains the range of the target value

sub: Callable = <built-in function sub>

xor: Callable = <built-in function xor>

class fhirpath.enums.SortOrderType(*value*)

Bases: enum.Enum

ASC: str = 'asc'

DESC: str = 'desc'

class fhirpath.enums.TermMatchType(*value*)

Bases: enum.Enum

ENDWITH: str = 'ENDWITH'

EXACT: str = 'EXACT'

FULLTEXT: str = 'FULLTEXT'

STARTWITH: str = 'STARTWITH'

class fhirpath.enums.WhereConstraintType(*value*)

Bases: enum.Enum

T1: str = 'T1'

T2: `str = 'T2'`

T3: `str = 'T3'`

`fhirpath.enums.ap(a: Any, b: Any) → Any`

approximately the value for the parameter in the resource is approximately the same to the provided value. Note that the recommended value for the approximation is 10% of the stated value (or for a date, 10% of the gap between now and the date), but systems may choose other values where appropriate

`fhirpath.enums.eb(a: Any, b: Any) → Any`

ends-before the value for the parameter in the resource ends before the provided value the range of the search value does overlap not with the range of the target value, and the range below the search value contains the range of the target value

`fhirpath.enums.sa(a: Any, b: Any) → Any`

starts-after the value for the parameter in the resource starts after the provided value the range of the search value does not overlap with the range of the target value, and the range above the search value contains the range of the target value

7.1.6 fhirpath.exceptions module

exception `fhirpath.exceptions.ConstraintNotSatisfied`

Bases: `zope.interface.exceptions.Invalid`

exception `fhirpath.exceptions.MultipleResultsFound`

Bases: `zope.interface.exceptions.Invalid`

exception `fhirpath.exceptions.NoResultFound`

Bases: `zope.interface.exceptions.Invalid`

exception `fhirpath.exceptions.ValidationError`

Bases: `fhirpath.exceptions.ConstraintNotSatisfied`

7.1.7 fhirpath.fhirpath module

Main module.

class `fhirpath.fhirpath.ClassInfo`

Bases: `object`

baseType: `fhirpath.types.TypeSpecifier`

static build_elements(`model_class: Type[FHIRAbstractModel]`) →
`List[fhirpath.fhirpath.ClassInfoElement]`

element: `List[fhirpath.fhirpath.ClassInfoElement]`

classmethod from_model(`model_class: Type[FHIRAbstractModel]`) → `ClassInfo`

get_elements() → `List[fhirpath.fhirpath.ClassInfoElement]`

name: `str`

namespace: `str`

class `fhirpath.fhirpath.ClassInfoElement`(`name: str, *, py_name: str, type_: fhirpath.types.TypeSpecifier, is_one_based: Optional[bool] = None, one_of_many_name: Optional[str] = None`)

Bases: `object`

build_simple_type_info() → `fhirpath.fhirpath.SimpleTypeInfo`

```

classmethod from_model_field(field: pydantic.fields.ModelField) → fhirpath.fhirpath.ClassInfoElement
isOneBased: Optional[bool]
name: str
type: fhirpath.types.TypeSpecifier
class fhirpath.fhirpath.FHIRPath(_obj: Any, predecessor: Optional[fhirpath.fhirpath.FHIRPath] = None)
    Bases: abc.ABC
    http://hl7.org/fhirpath/N1
abs()
    5.1.1. abs() : Integer | Decimal | Quantity Returns the absolute value of the input. When taking the absolute
    value of a quantity, the unit is unchanged.

    If the input collection is empty, the result is empty.

    If the input collection contains multiple items, the evaluation of the expression will end and signal an error
    to the calling environment.

    ``(-5).abs() // 5 (-5.5).abs() // 5.5 (-5.5 'mg').abs() // 5.5 'mg'``

all()
    5.1.3. all(criteria : expression) : Boolean Returns true if for every element in the input collection, criteria
    evaluates to true. Otherwise, the result is false. If the input collection is empty ({ }), the result is true.

    generalPractitioner.all($this is Practitioner)

allFalse()
    5.1.6. allFalse() : Boolean Takes a collection of Boolean values and returns true if all the items are false.
    If any items are true, the result is false. If the input is empty ({ }), the result is true.

    The following example returns true if none of the components of the Observation have a value greater than
    90 mm[Hg]: Observation.select(component.value > 90 'mm[Hg]').allFalse()

allTrue()
    5.1.4. allTrue() : Boolean Takes a collection of Boolean values and returns true if all the items are true. If
    any items are false, the result is false. If the input is empty ({ }), the result is true.

    The following example returns true if all of the components of the Observation have a value greater than
    90 mm[Hg]: Observation.select(component.value > 90 'mm[Hg]').allTrue()

anyFalse()
    5.1.7. anyFalse() : Boolean Takes a collection of Boolean values and returns true if any of the items are
    false. If all the items are true, or if the input is empty ({ }), the result is false.

    he following example returns true if any of the components of the Observation have a value that is not
    greater than 90 mm[Hg]: Observation.select(component.value > 90 'mm[Hg]').anyFalse()

anyTrue()
    5.1.5. anyTrue() : Boolean Takes a collection of Boolean values and returns true if any of the items are
    true. If all the items are false, or if the input is empty ({ }), the result is false.

    The following example returns true if any of the components of the Observation have a value greater than
    90 mm[Hg]: Observation.select(component.value > 90 'mm[Hg]').anyTrue()

as_(type_cls: Union[type, str])
    6.3.4. as(type : type specifier) The as() function is supported for backwards compatibility with previous
    implementations of FHIRPath. Just as with the as keyword, the type argument is an identifier that must
    resolve to the name of a type in a model. For implementations with compile-time typing, this requires
    special-case handling when processing the argument to treat is a type specifier rather than an identifier
    expression: ``Observation.component.where(value.as(Quantity) > 30 'mg')``

```

```
static build_fhir_abstract_type_info(klass: Type[FHIRAbstractModel], is_one_based: bool =
                                     True) → Union[fhirpath.fhirpath.ClassInfo,
                                     fhirpath.fhirpath.ListTypeInfo,
                                     fhirpath.fhirpath.TupleTypeInfo]
```

```
static build_type_info_from_el(element: Union[fhirpath.fhirpath.ClassInfoElement,
                                     fhirpath.fhirpath.TupleTypeInfoElement]) →
                                Union[fhirpath.fhirpath.ClassInfo, fhirpath.fhirpath.SimpleTypeInfo,
                                fhirpath.fhirpath.ListTypeInfo, fhirpath.fhirpath.TupleTypeInfo]
```

ceiling()

5.7.2. ceiling() : Integer Returns the first integer greater than or equal to the input.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 1.ceiling() // 1 1.1.ceiling() // 2 (-1.1).ceiling() // -1 ``
```

children()

5.8.1. children() : collection Returns a collection with all immediate child nodes of all items in the input collection. Note that the ordering of the children is undefined and using functions like first() on the result may return different results on different platforms.

combine()

5.4.2. combine(other : collection) : collection Merge the input and other collections into a single collection without eliminating duplicate values. Combining an empty collection with a non-empty collection will return the non-empty collection. There is no expectation of order in the resulting collection.

contained()

6.4.3. contains (containship) If the right operand is a collection with a single item, this operator returns true if the item is in the left operand using equality semantics. If the right-hand side of the operator is empty, the result is empty, if the left-hand side is empty, the result is false. This is the converse operation of in.

The following example returns true if the list of given names for the Patient has 'Joe' in it:

```
Patient.name.given contains 'Joe'
```

contains()

5.6.5. contains(substring : String) : Boolean Returns true when the given substring is a substring of the input string.

If substring is the empty string (''), the result is true.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abc'.contains('b') // true 'abc'.contains('bc') // true 'abc'.contains('d') // false ``
```

`` Note: The .contains() function described here is a string function that looks for a substring in a string. This is different than the contains operator, which is a list operator that looks for an element in a list. ``

```
static convert_and_cache_elements(elements: Union[List[fhirpath.fhirpath.ClassInfoElement],
                                     List[fhirpath.fhirpath.TupleTypeInfoElement]]) → None
```

convertsToBoolean()

5.5.2. convertsToBoolean() : Boolean If the input collection contains a single item, this function will return true if:

- the item is a Boolean

- the item is an Integer that is equal to one of the possible integer representations of Boolean values
- the item is a Decimal that is equal to one of the possible decimal representations of Boolean values
- the item is a String that is equal to one of the possible string representations of Boolean values

If the item is not one of the above types, or the item is a String, Integer, or Decimal, but is not equal to one of the possible values convertible to a Boolean, the result is false.

Possible values for Integer, Decimal, and String are described in the `toBoolean()` function.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToDate()

5.5.4. `convertsToDate()` : Boolean If the input collection contains a single item, this function will return true if:

- the item is a Date
- the item is a DateTime
- the item is a String and is convertible to a Date

If the item is not one of the above types, or is not convertible to a Date (using the format YYYY-MM-DD), the result is false.

If the item contains a partial date (e.g. '2012-01'), the result is a partial date.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToDateTime()

5.5.5. `convertsToDateTime()` : Boolean If the input collection contains a single item, this function will return true if:

- the item is a DateTime
- the item is a Date
- the item is a String and is convertible to a DateTime

If the item is not one of the above types, or is not convertible to a DateTime (using the format YYYY-MM-DDThh:mm:ss.fff(+-)hh:mm), the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToDecimal()

5.5.6. `convertsToDecimal()` : Boolean If the input collection contains a single item, this function will true if:

- the item is an Integer or Decimal
- the item is a String and is convertible to a Decimal
- the item is a Boolean

If the item is not one of the above types, or is not convertible to a Decimal (using the regex format `(+|-)?d+(.d+)?`), the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToInteger()

5.5.3 `convertsToInteger()` : Boolean If the input collection contains a single item, this function will return true if:

- the item is an Integer

- the item is a String and is convertible to an Integer
- the item is a Boolean

If the item is not one of the above types, or the item is a String, but is not convertible to an Integer (using the regex format `(+|-)?d+`), the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToQuantity()

5.5.7. `convertsToQuantity([unit : String]) : Boolean` If the input collection contains a single item, this function will return true if:

- the item is an Integer, Decimal, or Quantity
- the item is a String that is convertible to a Quantity
- the item is a Boolean

If the item is not one of the above types, or is not convertible to a Quantity using the following regex format: `(?'value'(\+|-)?\d+(\.\d+)?)\s*(?'unit'[^']+)|('time'[a-zA-Z]+))?` then the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty. @see <https://www.hl7.org/fhirpath/#convertstoquantityunit-string-boolean>

convertsToString()

5.5.8. `convertsToString() : String` If the input collection contains a single item, this function will return true if:

- the item is a String
- the item is an Integer, Decimal, Date, Time, or DateTime
- the item is a Boolean
- the item is a Quantity

If the item is not one of the above types, the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

convertsToTime()

5.5.9. `convertsToTime() : Boolean` If the input collection contains a single item, this function will return true if:

- the item is a Time
- the item is a String and is convertible to a Time

If the item is not one of the above types, or is not convertible to a Time (using the format `hh:mm:ss.fff(+|-)hh:mm`), the result is false.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

count() → int

5.1.10. `count() : Integer` Returns the integer count of the number of items in the input collection. Returns 0 when the input collection is empty.

descendants()

5.8.2. `descendants() : collection` Returns a collection with all descendant nodes of all items in the input collection. The result does not include the nodes in the input collection themselves. This function is a

shorthand for `repeat(children())`. Note that the ordering of the children is undefined and using functions like `first()` on the result may return different results on different platforms.

note`` Note: Many of these functions will result in a set of nodes of different underlying types. It may be necessary to use `ofType()` as described in the previous section to maintain type safety. See Type safety and strict evaluation for more information about type safe use of FHIRPath expressions. ``

distinct()

5.1.11. `distinct()` : collection Returns a collection containing only the unique items in the input collection. To determine whether two items are the same, the `= (Equals) (=)` operator is used, as defined below.

If the input collection is empty (`{ }`), the result is empty.

Note that the order of elements in the input collection is not guaranteed to be preserved in the result.

The following example returns the distinct list of tags on the given Patient: `Patient.meta.tag.distinct()`

```
static element_from_predecessor(predecessor: fhirpath.fhirpath.FHIRPath, prop_name: str) →
    Union[fhirpath.fhirpath.ClassInfoElement,
    fhirpath.fhirpath.TupleTypeInfoElement]
```

empty() → bool

5.1.1. `empty()` : Boolean Returns true if the input collection is empty (`{ }`) and false otherwise.

endsWith()

5.6.4. `endsWith(suffix : String)` : Boolean Returns true when the input string ends with the given suffix.

If suffix is the empty string (`''`), the result is true.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abcdefg'.endsWith('efg') // true 'abcdefg'.endsWith('abc') // false ``
```

exclude()

5.3.9. `exclude(other: collection)` : collection Returns the set of elements that are not in the other collection. Duplicate items will not be eliminated by this function, and order will be preserved. e.g. `(1 | 2 | 3).exclude(2)` returns `(1 | 3)`.

exists()

5.1.2. `exists([criteria : expression])` : Boolean Returns true if the collection has any elements, and false otherwise. This is the opposite of `empty()`, and as such is a shorthand for `empty().not()`. If the input collection is empty (`{ }`), the result is false.

The function can also take an optional criteria to be applied to the collection prior to the determination of the exists. In this case, the function is shorthand for `where(criteria).exists()`.

exp()

5.7.3. `exp()` : Decimal Returns e raised to the power of the input.

If the input collection contains an Integer, it will be implicitly converted to a Decimal and the result will be a Decimal.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 0.exp() // 1.0 (-0.0).exp() // 1.0 ``
```

first()

5.3.3. `first()` : collection Returns a collection containing only the first item in the input collection. This function is equivalent to `item[0]`, so it will return an empty collection if the input collection has no items.

floor()

5.7.4. `floor()` : Integer Returns the first integer less than or equal to the input.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 1.floor() // 1 2.1.floor() // 2 (-2.1).floor() // -3 ``
```

get_type()

FHIRPath supports reflection to provide the ability for expressions to access type information describing the structure of values. The `type()` function returns the type information for each element of the input collection, using one of the following concrete subtypes of `TypeInfo`:

iif()

5.5.1. `iif(criterion: expression, true-result: collection [, otherwise-result: collection])` : collection

The `iif` function in FHIRPath is an immediate if, also known as a conditional operator (such as C's `?` : operator). The criterion expression is expected to evaluate to a Boolean. If criterion is true, the function returns the value of the true-result argument.

If criterion is false or an empty collection, the function returns otherwise-result, unless the optional otherwise-result is not given, in which case the function returns an empty collection.

Note that short-circuit behavior is expected in this function. In other words, true-result should only be evaluated if the criterion evaluates to true, and otherwise-result should only be evaluated otherwise. For implementations, this means delaying evaluation of the arguments.

in_()

6.4.2. `in_ (membership)` If the left operand is a collection with a single item, this operator returns true if the item is in the right operand using equality semantics. If the left-hand side of the operator is empty, the result is empty, if the right-hand side is empty, the result is false. If the left operand has multiple items, an exception is thrown.

The following example returns true if 'Joe' is in the list of given names for the Patient: `` 'Joe' in Patient.name.given ``

indexOf()

5.6.1. `indexOf(substring : String)` : Integer Returns the 0-based index of the first position substring is found in the input string, or -1 if it is not found.

If substring is an empty string (`''`), the function returns 0.

If the input or substring is empty (`{ }`), the result is empty (`{ }`).

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment. `` 'abcdefg'.indexOf('bc') // 1 'abcdefg'.indexOf('x') // -1 'abcdefg'.indexOf('abcdefg') // 0 ``

intersect()

5.3.8. `intersect(other: collection)` : collection Returns the set of elements that are in both collections. Duplicate items will be eliminated by this function. Order of items is not guaranteed to be preserved in the result of this function.

isDistinct()

5.1.12. `isDistinct()` : Boolean Returns true if all the items in the input collection are distinct. To determine whether two items are distinct, the `=` (Equals) (`=`) operator is used, as defined below.

Conceptually, this function is shorthand for a comparison of the `count()` of the input collection against the `count()` of the `distinct()` of the input collection: `X.count() = X.distinct().count()` This means that if the input collection is empty (`{ }`), the result is true.

is_(*type_cls: Union[type, str]*)

6.3.2. `is(type : type specifier)` The `is()` function is supported for backwards compatibility with previous implementations of FHIRPath. Just as with the `is` keyword, the `type` argument is an identifier that must resolve to the name of a type in a model. For implementations with compile-time typing, this requires special-case handling when processing the argument to treat it as a type specifier rather than an identifier expression: `Patient.contained.all($this.is(Patient) implies age > 10)`

last()

5.3.4. `last() : collection` Returns a collection containing only the last item in the input collection. Will return an empty collection if the input collection has no items.

length()

5.6.11. `length() : Integer` Returns the length of the input string. If the input collection is empty (`{ }`), the result is empty.

ln()

5.7.5. `ln() : Decimal` Returns the natural logarithm of the input (i.e. the logarithm base *e*).

When used with an Integer, it will be implicitly converted to a Decimal.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment. `` 1.ln() // 0.0 1.0.ln() // 0.0 ``

log()

5.7.6. `log(base : Decimal) : Decimal` Returns the logarithm base *base* of the input number.

When used with Integers, the arguments will be implicitly converted to Decimal.

If *base* is empty, the result is empty.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

`` 16.log(2) // 4.0 100.0.log(10.0) // 2.0 ``

lower()

5.6.7. `lower() : String` Returns the input string with all characters converted to lower case.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

`` 'ABCDEFGH'.lower() // 'abcdefgh' 'aBcDEFG'.lower() // 'abcdefgh' ``

matches()

5.6.9. `matches(regex : String) : Boolean` Returns true when the value matches the given regular expression. Regular expressions should function consistently, regardless of any culture- and locale-specific settings in the environment, should be case-sensitive, use 'single line' mode and allow Unicode characters.

If the input collection or *regex* are empty, the result is empty (`{ }`).

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

now()

`now()` : `DateTime` Returns the current date and time, including timezone offset.

ofType(*type_cls*: *Union[type, str]*)

5.2.4. `ofType(type : type specifier) : collection` Returns a collection that contains all items in the input collection that are of the given type or a subclass thereof. If the input collection is empty (`{ }`), the result is empty. The type argument is an identifier that must resolve to the name of a type in a model. For implementations with compile-time typing, this requires special-case handling when processing the argument to treat it as type specifier rather than an identifier expression: `Bundle.entry.resource.ofType(Patient)`

In the above example, the symbol `Patient` must be treated as a type identifier rather than a reference to a `Patient` in context.

power()

5.7.7. `power(exponent : Integer | Decimal) : Integer | Decimal` Raises a number to the exponent power. If this function is used with Integers, the result is an Integer. If the function is used with Decimals, the result is a Decimal. If the function is used with a mixture of Integer and Decimal, the Integer is implicitly converted to a Decimal and the result is a Decimal.

If the power cannot be represented (such as the -1 raised to the 0.5), the result is empty.

If the input is empty, or exponent is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 2.power(3) // 8 2.5.power(2) // 6.25 (-1).power(0.5) // empty ({ }) ``
```

repeat()

5.2.3. `repeat(projection: expression) : collection` A version of `select` that will repeat the projection and add it to the output collection, as long as the projection yields new items (as determined by the `=` (Equals) (`=`) operator).

This function can be used to traverse a tree and selecting only specific children: `ValueSet.expansion.repeat(contains)`

Will repeat finding children called `contains`, until no new nodes are found. `Questionnaire.repeat(item)`

Will repeat finding children called `item`, until no new nodes are found. Note that this is slightly different from: `Questionnaire.descendants().select(item)`

which would find any descendants called `item`, not just the ones nested inside other item elements. The order of items returned by the `repeat()` function is undefined.

replace()

5.6.8. `replace(pattern : String, substitution : String) : String` Returns the input string with all instances of `pattern` replaced with `substitution`. If the substitution is the empty string (`''`), instances of `pattern` are removed from the result. If `pattern` is the empty string (`''`), every character in the input string is surrounded by the substitution, e.g. `'abc'.replace('','x')` becomes `'xaxbxcx'`.

If the input collection, `pattern`, or `substitution` are empty, the result is empty (`{ }`).

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abcdefg'.replace('cde', '123') // 'ab123fg' 'abcdefg'.replace('cde', '') // 'abfg' 'abc'.replace('','x') // 'xaxbxcx' ``
```

replaceMatches()

5.6.10. replaceMatches(regex [String, substitution: String])[String] Matches the input using the regular expression in regex and replaces each match with the substitution string. The substitution may refer to identified match groups in the regular expression.

If the input collection, regex, or substitution are empty, the result is empty (`{ }`).

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

This example of `replaceMatches()` will convert a string with a date formatted as MM/dd/yy to dd-MM-yy:

```
'11/30/1972'.replace('\b(?:<month>\d{1,2})/
(?:<day>\d{1,2})/(?:<year>\d{2,4})\b', '${day}-${month}-${year}')
```

`` Note: Platforms will typically use native regular expression implementations. These are typically fairly similar, but there will always be small differences. As such, FHIRPath does not prescribe a particular dialect, but recommends the use of the [PCRE] flavor as the dialect most likely to be broadly supported and understood.``

round()

5.7.8. `round([precision : Integer])` : Decimal Rounds the decimal to the nearest whole number using a traditional round (i.e. 0.5 or higher will round to 1). If specified, the precision argument determines the decimal place at which the rounding will occur. If not specified, the rounding will default to 0 decimal places.

If specified, the number of digits of precision must be ≥ 0 or the evaluation will end and signal an error to the calling environment.

If the input collection contains a single item of type Integer, it will be implicitly converted to a Decimal.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

`` 1.round() // 1 3.14159.round(3) // 3.142 ``

select()

5.2.2. `select(projection: expression)` : collection Evaluates the projection expression for each item in the input collection. The result of each evaluation is added to the output collection. If the evaluation results in a collection with multiple items, all items are added to the output collection (collections resulting from evaluation of projection are flattened). This means that if the evaluation for an element results in the empty collection (`{ }`), no element is added to the result, and that if the input collection is empty (`{ }`), the result is empty as well. `Bundle.entry.select(resource as Patient)`

This example results in a collection with only the patient resources from the bundle. `Bundle.entry.select((resource as Patient).telecom.where(system = 'phone'))`

This example results in a collection with all the telecom elements with system of phone for all the patients in the bundle. `Patient.name.where(use = 'usual').select(given.first() + ' ' + family)`
This example returns a collection containing, for each “usual” name for the Patient, the concatenation of the first given and family names.

single()

5.3.2. `single()` : collection Will return the single item in the input if there is just one item. If the input collection is empty (`{ }`), the result is empty. If there are multiple items, an error is signaled to the evaluation environment. This function is useful for ensuring that an error is returned if an assumption about cardinality is violated at run-time.

The following example returns the name of the Patient if there is one. If there are no names, an empty collection, and if there are multiple names, an error is signaled to the evaluation environment: `Patient.name.single()`

skip(*num*: *int*)

5.3.6. `skip(num : Integer) : collection` Returns a collection containing all but the first `num` items in the input collection. Will return an empty collection if there are no items remaining after the indicated number of items have been skipped, or if the input collection is empty. If `num` is less than or equal to zero, the input collection is simply returned.

sqrt()

5.7.9. `sqrt() : Decimal` Returns the square root of the input number as a Decimal.

If the square root cannot be represented (such as the square root of -1), the result is empty.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

Note that this function is equivalent to raising a number of the power of 0.5 using the `power()` function.

```
`` 81.sqrt() // 9.0 (-1).sqrt() // empty ``
```

startsWith()

5.6.3. `startsWith(prefix : String) : Boolean` Returns true when the input string starts with the given prefix.

If prefix is the empty string (''), the result is true.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abcdefg'.startsWith('abc') // true 'abcdefg'.startsWith('xyz') // false ``
```

subsetOf()

5.1.8. `subsetOf(other : collection) : Boolean` Returns true if all items in the input collection are members of the collection passed as the other argument. Membership is determined using the = (Equals) (=) operation.

Conceptually, this function is evaluated by testing each element in the input collection for membership in the other collection, with a default of true. This means that if the input collection is empty ({ }), the result is true, otherwise if the other collection is empty ({ }), the result is false.

The following example returns true if the tags defined in any contained resource are a subset of the tags defined in the MedicationRequest resource: `MedicationRequest.contained.meta.tag.subsetOf(MedicationRequest.meta.tag)`

substring()

5.6.2. `substring(start : Integer [, length : Integer]) : String` Returns the part of the string starting at position `start` (zero-based). If `length` is given, will return at most `length` number of characters from the input string.

If `start` lies outside the length of the string, the function returns empty ({ }). If there are less remaining characters in the string than indicated by `length`, the function returns just the remaining characters.

If the input or `start` is empty, the result is empty.

If an empty `length` is provided, the behavior is the same as if `length` had not been provided.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abcdefg'.substring(3) // 'defg' 'abcdefg'.substring(1, 2) // 'bc' 'abcdefg'.substring(6, 2) // 'g'
'abcdefg'.substring(7, 1) // { } ``
```

supersetOf()

5.1.9. **supersetOf(other : collection) : Boolean** Returns true if all items in the collection passed as the other argument are members of the input collection. Membership is determined using the = (Equals) (=) operation.

Conceptually, this function is evaluated by testing each element in the other collection for membership in the input collection, with a default of true. This means that if the other collection is empty (`{ }`), the result is true, otherwise if the input collection is empty (`{ }`), the result is false.

The following example returns true if the tags defined in any contained resource are a superset of the tags defined in the MedicationRequest resource: `MedicationRequest.contained.meta.tag.supersetOf(MedicationRequest.meta.tag)`

tail()

5.3.5. **tail() : collection** Returns a collection containing all but the first item in the input collection. Will return an empty collection if the input collection has no items, or only one item.

take(num: int)

5.3.7. **take(num : Integer) : collection** Returns a collection containing the first num items in the input collection, or less if there are less than num items. If num is less than or equal to 0, or if the input collection is empty (`{ }`), take returns an empty collection.

timeOfDay()

timeOfDay() : Time Returns the current time.

toBoolean()

5.5.2 **toBoolean() : Boolean** If the input collection contains a single item, this function will return a single boolean if:

- the item is a Boolean
- the item is an Integer and is equal to one of the possible integer representations of Boolean values
- the item is a Decimal that is equal to one of the possible decimal representations of Boolean values
- the item is a String that is equal to one of the possible string representations of Boolean values

If the item is not one the above types, or the item is a String, Integer, or Decimal, but is not equal to one of the possible values convertible to a Boolean, the result is empty. @see: <https://www.hl7.org/fhirpath/#boolean-conversion-functions>

toChars()

5.6.12. **toChars() : collection** Returns the list of characters in the input string. If the input collection is empty (`{ }`), the result is empty. `` 'abc'.toChars() // { 'a', 'b', 'c' } ``

toDate()

5.5.4 **toDate() : Date** If the input collection contains a single item, this function will return a single date if:

- the item is a Date
- the item is a DateTime
- the item is a String and is convertible to a Date

If the item is not one of the above types, the result is empty.

If the item is a String, but the string is not convertible to a Date (using the format YYYY-MM-DD), the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

toDateTime()

5.5.5 **toDateTime() : DateTime** If the input collection contains a single item, this function will return a single datetime if:

- the item is a DateTime

- the item is a Date, in which case the result is a DateTime with the year, month, and day of the Date, and the time components empty (not set to zero)
- the item is a String and is convertible to a DateTime

If the item is not one of the above types, the result is empty. If the item is a String, but the string is not convertible to a DateTime (using the format YYYY-MM-DDThh:mm:ss.fff(+|-)hh:mm), the result is empty.

If the item contains a partial datetime (e.g. '2012-01-01T10:00'), the result is a partial datetime.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

toDecimal()

5.5.6. toDecimal() : Decimal If the input collection contains a single item, this function will return a single decimal if:

- the item is an Integer or Decimal
- the item is a String and is convertible to a Decimal
- the item is a Boolean, where true results in a 1.0 and false results in a 0.0.

If the item is not one of the above types, the result is empty.

If the item is a String, but the string is not convertible to a Decimal (using the regex format (+|-)?d+(.d+)?), the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

toInteger()

5.5.3 toInteger() : Integer If the input collection contains a single item, this function will return a single integer if:

- the item is an Integer
- the item is a String and is convertible to an integer
- the item is a Boolean, where true results in a 1 and false results in a 0.

If the item is not one the above types, the result is empty.

If the item is a String, but the string is not convertible to an integer (using the regex format (\+|-)?\d+), the result is empty. If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment. If the input collection is empty, the result is empty.

toQuantity()

5.5.7. toQuantity([unit : String]) : Quantity If the input collection contains a single item, this function will return a single quantity if:

- the item is an Integer, or Decimal, where the resulting quantity will have the default unit ('1')
- the item is a Quantity
- the item is a String and is convertible to a Quantity
- the item is a Boolean, where true results in the quantity 1.0 '1', and false results in the quantity 0.0 '1'

If the item is not one of the above types, the result is empty. If the item is a String, but the string is not convertible

to a Quantity using the following regex format: (? 'value' (\+|-)?\d+(\.\d+)?)\s*('(? 'unit' [^']+)' | '(? 'time' [a-zA-Z]+)')? then the result is empty. For example, the following are valid quantity strings: '4 days' '10 'mg[Hg]''

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty. @see <https://www.hl7.org/fhirpath/#toquantityunit-string-quantity>

toString()

5.5.8. toString() : String If the input collection contains a single item, this function will return a single String if:

- the item in the input collection is a String
- the item in the input collection is an Integer, Decimal, Date, Time, DateTime, or Quantity the output will contain its String representation
- the item is a Boolean, where true results in 'true' and false in 'false'.

If the item is not one of the above types, the result is false. @see <https://www.hl7.org/fhirpath/#tostring-string>

toTime()

5.5.9. toTime() : Time If the input collection contains a single item, this function will return a single time if:

- the item is a Time
- the item is a String and is convertible to a Time

If the item is not one of the above types, the result is empty.

If the item is a String, but the string is not convertible to a Time (using the format hh:mm:ss.fff(+|-)hh:mm), the result is empty.

If the item contains a partial time (e.g. '10:00'), the result is a partial time.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

If the input collection is empty, the result is empty.

today()

today() : Date Returns the current date.

trace()

5.9.1. trace(name : String [, projection: Expression]) : collection Adds a String representation of the input collection to the diagnostic log, using the name argument as the name in the log. This log should be made available to the user in some appropriate fashion. Does not change the input, so returns the input collection as output.

If the projection argument is used, the trace would log the result of evaluating the project expression on the input, but still return the input to the trace function unchanged.

contained.where(criteria).trace('unmatched', id).empty() The above example traces only the id elements of the result of the where.

truncate()

5.7.10. truncate() : Integer Returns the integer portion of the input.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
^^ 101.truncate() // 101 1.00000001.truncate() // 1 (-1.56).truncate() // -1 ^^
```

union()

5.4.1. union(other : collection) Merge the two collections into a single collection, eliminating any duplicate values (using = (Equals) (=) to determine equality). There is no expectation of order in the resulting collection.

In other words, this function returns the distinct list of elements from both inputs. For example, consider two lists of integers A: 1, 1, 2, 3 and B: 2, 3: A union B // 1, 2, 3 A union { } // 1, 2, 3

This function can also be invoked using the `|` operator. `a.union(b)` is synonymous with `a | b`

upper()

5.6.6. `upper()` : String Returns the input string with all characters converted to upper case.

If the input collection is empty, the result is empty.

If the input collection contains multiple items, the evaluation of the expression will end and signal an error to the calling environment.

```
`` 'abcdefg'.upper() // 'ABCDEFGG' 'AbCdefg'.upper() // 'ABCDEFGG' ``
```

where()

5.2.1. `where(criteria : expression) : collection` Returns a collection containing only those elements in the input collection for which the stated criteria expression evaluates to true. Elements for which the expression evaluates to false or empty (`{ }`) are not included in the result

If the input collection is empty (`{ }`), the result is empty. If the result of evaluating the condition is other than a single boolean value, the evaluation will end and signal an error to the calling environment, consistent with singleton evaluation of collections behavior.

The following example returns the list of telecom elements that have a use element with the value of 'official': `Patient.telecom.where(use = 'official')`

class fhirpath.fhirpath.ListTypeInfo

Bases: object

For collection types, the result is a ListTypeInfo: `ListTypeInfo { elementType: TypeSpecifier }`

For example: `Patient.address.type()`

Results in: `{ListTypeInfo { elementType: 'FHIR.Address' }}`

elementType: `fhirpath.types.TypeSpecifier`

classmethod from_specifier(*specifier*) → *fhirpath.fhirpath.ListTypeInfo*

get_elements() → Union[List[*fhirpath.fhirpath.ClassInfoElement*],
List[*fhirpath.fhirpath.TupleTypeInfoElement*]]

class fhirpath.fhirpath.SimpleTypeInfo

Bases: object

For primitive types such as String and Integer, the result is a SimpleTypeInfo:

baseType: `fhirpath.types.TypeSpecifier`

classmethod from_type_specifier(*specifier: fhirpath.types.TypeSpecifier*) →
fhirpath.fhirpath.SimpleTypeInfo

name: `str`

namespace: `str`

class fhirpath.fhirpath.TupleTypeInfo

Bases: object

Anonymous types are structured types that have no associated name, only the elements of the structure. For example, in FHIR, the `Patient.contact` element has multiple sub-elements, but is not explicitly named. For types such as this, the result is a TupleTypeInfo: @see <http://hl7.org/fhirpath/N1/#anonymous-types>

static build_elements(*model_class: Type[FHIRAbstractModel]*) →
List[*fhirpath.fhirpath.TupleTypeInfoElement*]

element: List[*fhirpath.fhirpath.TupleTypeInfoElement*]

classmethod from_model(*model_class: Type[FHIRAbstractModel]*) → *TupleTypeInfo*

```

    get_elements() → List[fhirpath.fhirpath.TupleTypeInfoElement]
class fhirpath.fhirpath.TupleTypeInfoElement(name: str, *, py_name: str, type_:
    fhirpath.types.TypeSpecifier, is_one_based:
    Optional[bool] = None, one_of_many_name:
    Optional[str] = None)

Bases: fhirpath.fhirpath.ClassInfoElement

isOneBased: Optional[bool]

name: str

type: fhirpath.types.TypeSpecifier

fhirpath.fhirpath.collection_type_required(func: Callable)

```

7.1.8 fhirpath.model module

```

class fhirpath.model.Model
    Bases: object

    static create(resource_type: str, fhir_release: fhirpath.enums.FHIR_VERSION =
        <FHIR_VERSION.DEFAULT: 'R4'>)

class fhirpath.model.ModelFactory(name, bases, attrs, **kwargs)
    Bases: type

    FHIR Model factory

    add_to_class(name, value)

```

7.1.9 fhirpath.query module

```

class fhirpath.query.AsyncQueryResult(query: fhirpath.query.Query, engine: Engine, unrestricted: bool =
    False)

Bases: fhirpath.query.QueryResult

async count()
    Returns the integer count of the number of items in the input collection. Returns 0 when the input collection
    is empty.

async empty()
    Returns true if the input collection is empty ({ }) and false otherwise.

async fetchall()

async first()

async single()

fhirpath.query.Q_(resource: Optional[Union[str, List[str]]] = None, engine=None)

class fhirpath.query.Query(fhir_release: fhirpath.enums.FHIR_VERSION, from_:
    fhirpath.fql.types.FromClause, select: fhirpath.fql.types.SelectClause, element:
    fhirpath.fql.types.ElementClause, where: fhirpath.fql.types.WhereClause, sort:
    fhirpath.fql.types.SortClause, limit: fhirpath.fql.types.LimitClause)

Bases: abc.ABC

clone() → fhirpath.query.Query

```

classmethod **from_builder**(*builder*: [fhirpath.query.QueryBuilder](#)) → [fhirpath.query.Query](#)
Create Query object from QueryBuilder. Kind of reverse process

get_element() → [fhirpath.fql.types.ElementClause](#)

get_from() → [fhirpath.fql.types.FromClause](#)

get_limit() → [fhirpath.fql.types.LimitClause](#)

get_select() → [fhirpath.fql.types.SelectClause](#)

get_sort() → [fhirpath.fql.types.SortClause](#)

get_where() → [fhirpath.fql.types.WhereClause](#)

class [fhirpath.query.QueryBuilder](#)(*engine*: *Optional[Engine]* = *None*)

Bases: [abc.ABC](#)

bind(*engine*: *Engine*)

clone() → [fhirpath.query.QueryBuilder](#)

element(**args*, ***kwargs*)

finalize(*engine*: *Optional[Engine]* = *None*)

from_(**args*, ***kwargs*)

get_query() → [fhirpath.query.Query](#)

limit(**args*, ***kwargs*)

select(**args*, ***kwargs*)

sort(**args*, ***kwargs*)

where(**args*, ***kwargs*)

class [fhirpath.query.QueryResult](#)(*query*: [fhirpath.query.Query](#), *engine*: *Engine*, *unrestricted*: *bool* = *False*)

Bases: [abc.ABC](#)

OFF__getitem__(*key*)

Lazy loading es results with negative index support. We store the results in buckets of what the bulk size is. This is so you can skip around in the indexes without needing to load all the data. Example(all zero based indexing here remember):

```
(525 results with bulk size 50)
- self[0]: 0 bucket, 0 item
- self[10]: 0 bucket, 10 item
- self[50]: 50 bucket: 0 item
- self[55]: 50 bucket: 5 item
- self[352]: 350 bucket: 2 item
- self[-1]: 500 bucket: 24 item
- self[-2]: 500 bucket: 23 item
- self[-55]: 450 bucket: 19 item
```

count() → *int*

Returns the integer count of the number of items in the input collection. Returns 0 when the input collection is empty.

count_raw()

Returns [EngineResult](#)

empty() → *bool*

Returns true if the input collection is empty ({ }) and false otherwise.

fetchall()

first()

Returns a collection containing only the first item in the input collection. This function is equivalent to `item(0)`, so it will return an empty collection if the input collection has no items.

last()

Returns a collection containing only the last item in the input collection. Will return an empty collection if the input collection has no items.

single()

Will return the single item in the input if there is just one item. If the input collection is empty (`{ }`), the result is empty. If there are multiple items, an error is signaled to the evaluation environment. This operation is useful for ensuring that an error is returned if an assumption about cardinality is violated at run-time.

skip(*num: int*)

Returns a collection containing all but the first *num* items in the input collection. Will return an empty collection if there are no items remaining after the indicated number of items have been skipped, or if the input collection is empty. If *num* is less than or equal to zero, the input collection is simply returned.

tail()

Returns a collection containing all but the first item in the input collection. Will return an empty collection if the input collection has no items, or only one item.

take(*num: int*)

Returns a collection containing the first *num* items in the input collection, or less if there are less than *num* items. If *num* is less than or equal to 0, or if the input collection is empty (`{ }`), `take` returns an empty collection.

7.1.10 fhirpath.search module

class `fhirpath.search.AsyncSearch`(*context: fhirpath.search.SearchContext, query_string=None, params=None*)

Bases: `fhirpath.search.Search`

class `fhirpath.search.Search`(*context: fhirpath.search.SearchContext, query_string=None, params=None*)

Bases: `object`

add_term(*normalized_data, terms_container*)

attach_elements_terms(*builder*)

attach_limit_terms(*builder*)

attach_sort_terms(*builder*)

attach_summary_terms(*builder*)

attach_where_terms(*builder*)

build() $\rightarrow$ `fhirpath.query.QueryResult`

Create QueryBuilder from search query string

create_address_term(*path_, param_value, modifier*)

create_codeableconcept_term(*path_, param_value, modifier*)

create_coding_term(*path_, param_value, modifier*)

create_contactpoint_term(*path_, param_value, modifier*)

create_exists_term(*path_, param_value, modifier*)

```
create_humanname_term(path_, param_value, modifier)
create_identifier_term(path_, param_value, modifier)
create_money_term(path_, param_value, modifier)
create_period_term(path_, param_value, modifier)
create_quantity_term(path_, param_value, modifier)
create_reference_term(path_, param_value, modifier)
create_term(path_, value, modifier)
classmethod from_params(context, params)
classmethod from_query_string(query_string)
has() → List[Tuple[fhirpath.fhirspec.spec.SearchParameter, fhirpath.query.QueryResult]]
    This function handles the _has keyword.
include(main_query_result: fhirpath.engine.base.EngineResult) → List[fhirpath.query.QueryResult]
    This function handles the _include keyword.
static parse_query_string(query_string, allow_none=False)
    param:request param:allow_none
prepare_params(all_params)
    making search, sort, limit, params Result Parameters ~~~~~ _sort _count _include _revin-
    clude _summary _total _elements _contained _containedType
response(result, includes, as_json)
rev_include(main_query_result: fhirpath.engine.base.EngineResult) → List[fhirpath.query.QueryResult]
    This function handles the _revinclude keyword.
single_valued_address_term(path_, value, modifier)
single_valued_codeableconcept_term(path_, value, modifier)
single_valued_coding_term(path_, value, modifier)
single_valued_contactpoint_term(path_, value, modifier)
single_valued_humanname_term(path_, value, modifier)
single_valued_identifier_term(path_, value, modifier)
single_valued_money_term(path_, value, modifier)
single_valued_period_term(path_, value, modifier)
single_valued_quantity_term(path_, value, modifier)
single_valued_reference_term(path_, value, modifier)
validate()
static validate_normalized_value(param_name, param_value, modifier)
```

Parameters

- `param_name` –
- `param_value` –
- `modifier` –

```
static validate_params(context, query_string, params)
```

```

    validate_pre_term(operator_, path_, value, modifier)

class fhirpath.search.SearchContext(engine, resource_type, unrestricted=False, async_result=None)
    Bases: object
    async_result
    augment_with_types(resource_types: List[str])
    definitions
    engine
    get_parameters_definition(fhir_release: fhirpath.enums.FHIR_VERSION) →
        List[fhirpath.fhirspec.spec.ResourceSearchParameterDefinition]
    normalize_param(param_name, raw_value) → List[Tuple[fhirpath.fql.types.ElementPath, str,
        Optional[str]]]
    normalize_param_value(raw_value: Union[List, str], search_param:
        fhirpath.fhirspec.spec.SearchParameter)
    parse_composite_parameter_component(component, raw_value, param_def, modifier)
    resolve_path_context(search_param: fhirpath.fhirspec.spec.SearchParameter)
    resource_types
    search_params_intersection
    unrestricted

fhirpath.search.fhir_search(context: fhirpath.search.SearchContext, query_string: Optional[str] = None,
    params: Optional[Union[Dict[str, str], Tuple[Tuple[str, str]]]] = None,
    response_as_dict: bool = False)

fhirpath.search.has_escape_comma(val)

```

7.1.11 fhirpath.storage module

```

class fhirpath.storage.MemoryStorage
    Bases: collections.defaultdict
    delete(item)
    empty()
    exists(item)
    get(item, default=<NO_VALUE>)
    insert(item, value)
    total()

```

7.1.12 fhirpath.types module

©FHIR Data Primitive Types <https://www.hl7.org/fhir/datatypes.html#primitive>

class fhirpath.types.FhirBase64Binary

Bases: bytes

A stream of bytes, base64 encoded (RFC 4648) There is no specified upper limit to the size of a binary, but systems will have to impose some implementation based limit to the size they support. This should be clearly documented, though there is no computable for this at this time

XML Representation: xs:base64Binary JSON representation: A JSON string - base64 content

to_python() → bytes

class fhirpath.types.FhirBoolean

Bases: fhirpath.types.FhirPrimitiveType

XML Representation: xs:boolean, except that 0 and 1 are not valid values JSON representation: JSON boolean (true or false)

to_python() → bool

class fhirpath.types.FhirCanonical

Bases: fhirpath.types.FhirPrimitiveType

A URI that refers to a resource by its canonical URL (resources with a url property). The canonical type differs from a uri in that it has special meaning in this specification, and in that it may have a version appended, separated by a vertical bar (|). Note that the type canonical is not used for the actual canonical URLs that are the target of these references, but for the URIs that refer to them, and may have the version suffix in them. Like other URIs, elements of type canonical may also have #fragment references

XML Representation: xs:anyURI JSON representation: A JSON string - a canonical URL

to_python() → str

class fhirpath.types.FhirCode

Bases: fhirpath.types.FhirPrimitiveType

Indicates that the value is taken from a set of controlled strings defined elsewhere (see Using codes for further discussion). Technically, a code is restricted to a string which has at least one character and no leading or trailing whitespace, and where there is no whitespace other than single spaces in the contents

XML Representation: xs:token JSON representation: JSON string

to_python() → str

class fhirpath.types.FhirDate

Bases: fhirpath.types.FhirPrimitiveType

A date, or partial date (e.g. just year or year + month) as used in human communication. The format is YYYY, YYYY-MM, or YYYY-MM-DD, e.g. 2018, 1973-06, or 1905-08-23. There SHALL be no time zone. Dates SHALL be valid dates

XML Representation: union of xs:date, xs:gYearMonth, xs:gYear JSON representation: A JSON string - a union of xs:date, xs:gYearMonth, xs:gYear

to_python() → datetime.date

class fhirpath.types.FhirDateTime

Bases: fhirpath.types.FhirPrimitiveType

A date, date-time or partial date (e.g. just year or year + month) as used in human communication. The format is YYYY, YYYY-MM, YYYY-MM-DD or YYYY-MM-DDThh:mm:ss+zz:zz, e.g. 2018, 1973-06, 1905-08-23, 2015-02-07T13:28:17-05:00 or 2017-01-01T00:00:00.000Z. If hours and minutes are specified, a time zone SHALL be populated. Seconds must be provided due to schema type constraints but may be zero-filled and may be ignored at receiver discretion. Dates SHALL be valid dates. The time “24:00” is not allowed. Leap Seconds are allowed

XML Representation: union of xs:dateTime, xs:date, xs:gYearMonth, xs:gYear JSON representation: A JSON string - a union of xs:dateTime, xs:date, xs:gYearMonth, xs:gYear

to_python() → datetime.datetime

class fhirpath.types.FhirDecimal

Bases: fhirpath.types.FhirPrimitiveType

Rational numbers that have a decimal representation. The precision of the decimal value has significance:

- e.g. 0.010 is regarded as different to 0.01, and the original precision should be preserved
- Implementations SHALL handle decimal values in ways that preserve and respect the precision of the value as represented for presentation purposes
- Implementations are not required to perform calculations with these numbers differently, though they may choose to do so (i.e. preserve significance)
- In object code, implementations that might meet this constraint are GMP implementations or equivalents to Java BigDecimal that implement arbitrary precision, or a combination of a (64 bit) floating point value with a precision field
- Note that large and/or highly precise values are extremely rare in medicine. One element where highly precise decimals may be encountered is the Location coordinates. Irrespective of this, the limits documented in XML Schema apply

XML Representation: union of xs:decimal and xs:double (see below for limitations) JSON representation: A JSON number (see below for limitations)

to_python() → float

class fhirpath.types.FhirId

Bases: fhirpath.types.FhirPrimitiveType

Any combination of upper- or lower-case ASCII letters ('A'..'Z', and 'a'..'z'), numerals ('0'..'9'), '-' and '.', with a length limit of 64 characters. (This might be an integer, an un-prefixed OID, UUID or any other identifier pattern that meets these constraints.)

XML Representation: xs:string JSON representation: JSON string

to_python() → str

class fhirpath.types.FhirInstant

Bases: fhirpath.types.FhirPrimitiveType

An instant in time in the format YYYY-MM-DDThh:mm:ss.sss+zz:zz (e.g. 2015-02-07T13:28:17.239+02:00 or 2017-01-01T00:00:00Z). The time SHALL specified at least to the second and SHALL include a time zone. Note: This is intended for when precisely observed times are required (typically system logs etc.), and not human-reported times - for those, use date or dateTime (which can be as precise as instant, but is not required to be). instant is a more constrained dateTime

XML Representation: xs:dateTime JSON representation: A JSON string - an xs:dateTime

to_python() → datetime.datetime

class fhirpath.types.FhirInteger

Bases: fhirpath.types.FhirPrimitiveType

A signed integer in the range -2,147,483,648..2,147,483,647 (32-bit; for larger values, use decimal)

XML Representation: xs:int, except that leading 0 digits are not allowed JSON representation: JSON number (with no decimal point)

to_python() → int

class fhirpath.types.FhirMarkdown

Bases: `fhirpath.types.FhirPrimitiveType`

A FHIR string (see above) that may contain markdown syntax for optional processing by a markdown presentation engine, in the GFM extension of CommonMark format (see below)

About the markdown datatype: - This specification requires and uses the GFM (Github Flavored Markdown) extensions on CommonMark format - Note that GFM prohibits Raw HTML - Systems are not required to have markdown support, so the content of a string should be readable without markdown processing, per markdown philosophy - Markdown content SHALL NOT contain Unicode character points below 32, except for u0009 (horizontal tab), u0010 (carriage return) and u0013 (line feed) - Markdown is a string, and subject to the same rules (e.g. length limit) - Converting an element that has the type string to markdown in a later version of this FHIR specification is not considered a breaking change (neither is adding markdown as a choice to an optional element that already has a choice of data types)

XML Representation: xs:string JSON representation: JSON string

to_python() → str

class fhirpath.types.FhirOid

Bases: `fhirpath.types.FhirPrimitiveType`

An OID represented as a URI (RFC 3001); e.g. `urn:oid:1.2.3.4.5`

XML Representation: xs:anyURI JSON representation: JSON string - uri

to_python() → str

class fhirpath.types.FhirPositiveInt

Bases: `fhirpath.types.FhirPrimitiveType`

Any positive integer in the range 1..2,147,483,647

XML Representation: xs:positiveInteger JSON representation: JSON number

to_python() → int

class fhirpath.types.FhirString

Bases: `fhirpath.types.FhirPrimitiveType`

A sequence of Unicode characters Note that strings SHALL NOT exceed 1MB (1024*1024 characters) in size. Strings SHOULD not contain Unicode character points below 32, except for u0009 (horizontal tab), u0010 (carriage return) and u0013 (line feed). Leading and Trailing whitespace is allowed, but SHOULD be removed when using the XML format. Note: This means that a string that consists only of whitespace could be trimmed to nothing, which would be treated as an invalid element value. Therefore strings SHOULD always contain non-whitespace content

XML Representation: xs:string JSON representation: JSON String

to_python() → str

class fhirpath.types.FhirTime

Bases: `fhirpath.types.FhirPrimitiveType`

A time during the day, in the format hh:mm:ss. There is no date specified. Seconds must be provided due to schema type constraints but may be zero-filled and may be ignored at receiver discretion. The time "24:00" SHALL NOT be used. A time zone SHALL NOT be present. Times can be converted to a Duration since midnight.

XML Representation: xs:time JSON representation: A JSON string - an xs:time

to_python() → float

class fhirpath.types.FhirURI

Bases: `fhirpath.types.FhirPrimitiveType`

A Uniform Resource Identifier Reference (RFC 3986). Note: URIs are case sensitive. For UUID (`urn:uuid:53fefa32-fcbb-4ff8-8a92-55ee120877b7`) use all lowercase

XML Representation: xs:anyURI JSON representation: A JSON string - a URI

to_python() → str

class fhirpath.types.FhirURL

Bases: `fhirpath.types.FhirPrimitiveType`

A Uniform Resource Locator (RFC 1738). Note URLs are accessed directly using the specified protocol. Common URL protocols are `http{s}:`, `ftp:`, `mailto:` and `mllp:`, though many others are defined

XML Representation: xs:anyURI JSON representation: A JSON string - a URL

to_python() → str

class fhirpath.types.FhirUUID

Bases: `fhirpath.types.FhirPrimitiveType`

A UUID (aka GUID) represented as a URI (RFC 4122); e.g. `urn:uuid:c757873d-ec9a-4326-a141-556f43239520`

XML Representation: xs:anyURI JSON representation: JSON string - uri

to_python() → str

class fhirpath.types.FhirUnsignedInt

Bases: `fhirpath.types.FhirPrimitiveType`

Any non-negative integer in the range 0..2,147,483,647

XML Representation: xs:nonNegativeInteger JSON representation: JSON number

to_python() → int

7.1.13 fhirpath.utils module

class fhirpath.utils.BundleWrapper(*engine, result, includes: List, url: yarl.URL, bundle_type='searchset', *, base_url: Optional[yarl.URL] = None, init_data: Optional[Dict[str, Any]] = None*)

Bases: `object`

FHIR_REST_SERVER_PATH_PATTERN: Optional[Pattern] = None

attach_entry(*result, mode='match'*)

attach_links(*url, entries_count*)

static calculate_fhir_base_url(*url: yarl.URL*) → `yarl.URL`

<https://www.hl7.org/fhir/Bundle.html> Section: 2.36.4 Resource URL & Uniqueness rules in a bundle.
Section: 2.36.4.1 Resolving references in Bundles.

classmethod fhir_rest_server_path_pattern()

static init_data() → `Dict[str, Any]`

Initialized Bundle data

```
    json()
    make_link(relation, url, params=None)
    resolve_absolute_uri(relative_path: str) → yarl.URL

class fhirpath.utils.EmptyPathInfoContext
    Bases: object

    Empty PathInfoContext for start(*) path!

class fhirpath.utils.PathInfoContext(path: str, fhir_release: fhirpath.enums.FHIR_VERSION,
                                     prop_name: str, prop_original: str, type_name: str, type_class:
                                     Union[bool, AbstractBaseType, AbstractType, Primitive], type_field:
                                     ModelField, type_model_config: Type[BaseConfig], optional: bool,
                                     multiple: bool, type_is_primitive: bool, resource_type: str)

    Bases: object
    add_child(path)
    property children
    classmethod context_from_path(pathname: str, fhir_release: fhirpath.enums.FHIR_VERSION) →
        Union[fhirpath.utils.PathInfoContext,
        fhirpath.utils.EmptyPathInfoContext]
    is_root()
    property parent: fhirpath.utils.PathInfoContext
    validate_value(value)
        pydantic way to validate value

class fhirpath.utils.PathInfoContextProxy(context: fhirpath.utils.PathInfoContext)
    Bases: fhirpath.thirdparty.peewee.Proxy
    obj

fhirpath.utils.builder(func)
    Decorator for wrapper “builder” functions. These are functions on the Query class or other classes used for
    building queries which mutate the query and return self. To make the build functions immutable, this decorator
    is used which will deepcopy the current instance. This decorator will return the return value of the inner function
    or the new copy of the instance. The inner function does not need to return self.

fhirpath.utils.expand_path(path_: str) → str
    Path normalizer Supports: 1. Home Path expander 2. Package path discovery

fhirpath.utils.fallback_callable(*args, **kwargs)
    Always return None

fhirpath.utils.force_bytes(string: str, encoding: str = 'utf8', errors: str = 'strict') → bytes

fhirpath.utils.force_str(value: Any, allow_non_str: bool = True) → str

fhirpath.utils.get_local_timezone() → datetime.timezone

fhirpath.utils.import_string(dotted_path: str) → type
    Shameless hack from django utils, please don't mind!

fhirpath.utils.lookup_all_fhir_domain_resource_classes(fhir_release:
                                                        fhirpath.enums.FHIR_VERSION =
                                                        <FHIR_VERSION.DEFAULT: 'R4'>) →
                                                        Dict[str, str]
```

`fhirpath.utils.lookup_fhir_class(resource_type: str, fhir_release: fhirpath.enums.FHIR_VERSION = <FHIR_VERSION.DEFAULT: 'R4'>) → Type[FHIRAbstractModel]`

`fhirpath.utils.lookup_fhir_class_path(resource_type: str, cache: bool = True, fhir_release: fhirpath.enums.FHIR_VERSION = <FHIR_VERSION.DEFAULT: 'R4'>) → Optional[str]`

This function finds FHIR resource model class (from `fhir.resources`) and return dotted path string.

Parameters

- **resource_type** – the resource type name (required). i.e Organization
- **cache** – (default True) the flag which indicates should query fresh or serve from cache if available.
- **fhir_release** – FHIR Release (version) name. i.e FHIR_VERSION.STU3, FHIR_VERSION.R4

:return dotted full string path. i.e `fhir.resources.organization.Organization`

Example:

```
>>> from fhirpath.utils import lookup_fhir_class_path
>>> from zope.interface import Invalid
>>> dotted_path = lookup_fhir_class_path('Patient')
>>> 'fhir.resources.patient.Patient' == dotted_path
True
>>> dotted_path = lookup_fhir_class_path('FakeResource')
>>> dotted_path is None
True
```

`fhirpath.utils.proxy(obj)`
Making proxy of any object

`fhirpath.utils.reraise(klass, msg=None, callback=None, **kw)`
Reraise custom exception class

`fhirpath.utils.timestamp_local()` → `datetime.datetime`
Timezone aware datetime with local timezone offset

`fhirpath.utils.timestamp_utc()` → `datetime.datetime`
UTC datetime with timezone offset

`fhirpath.utils.unwrap_proxy(proxy_obj)`

7.1.14 fhirpath.version module

7.1.15 Module contents

Top-level package for fhirpath.

`fhirpath.get_version()`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

f

- [fhirpath](#), 65
- [fhirpath.connectors](#), 25
 - [fhirpath.connectors.connection](#), 23
 - [fhirpath.connectors.factory](#), 23
 - [fhirpath.connectors.interfaces](#), 24
 - [fhirpath.connectors.url](#), 24
- [fhirpath.constraints](#), 38
- [fhirpath.dialects](#), 27
 - [fhirpath.dialects.base](#), 25
 - [fhirpath.dialects.elasticsearch](#), 25
 - [fhirpath.dialects.postgres](#), 26
 - [fhirpath.dialects.sqlalchemy](#), 26
- [fhirpath.engine](#), 29
 - [fhirpath.engine.base](#), 27
 - [fhirpath.engine.es](#), 28
 - [fhirpath.engine.fhirbase](#), 29
- [fhirpath.enums](#), 38
- [fhirpath.exceptions](#), 40
- [fhirpath.fhirpath](#), 40
- [fhirpath.fhirspec](#), 32
 - [fhirpath.fhirspec.downloader](#), 30
 - [fhirpath.fhirspec.spec](#), 30
- [fhirpath.fql](#), 35
 - [fhirpath.fql.expressions](#), 32
 - [fhirpath.fql.types](#), 33
- [fhirpath.interfaces](#), 36
 - [fhirpath.interfaces.base](#), 36
 - [fhirpath.interfaces.connectors](#), 36
 - [fhirpath.interfaces.dialects](#), 36
 - [fhirpath.interfaces.engine](#), 36
 - [fhirpath.interfaces.fql](#), 36
- [fhirpath.model](#), 55
- [fhirpath.query](#), 55
- [fhirpath.search](#), 57
- [fhirpath.storage](#), 59
- [fhirpath.thirdparty](#), 37
 - [fhirpath.thirdparty.peewee](#), 36
 - [fhirpath.thirdparty.werkzeug](#), 37
- [fhirpath.types](#), 60
- [fhirpath.utils](#), 63
- [fhirpath.version](#), 65

A

- `abs()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `add()` (*fhirpath.engine.base.EngineResultBody* method), 27
- `add()` (*fhirpath.engine.EngineResultBody* method), 30
- `add_child()` (*fhirpath.utils.PathInfoContext* method), 64
- `add_term()` (*fhirpath.search.Search* method), 57
- `add_to_class()` (*fhirpath.model.ModelFactory* method), 55
- `ALL` (*fhirpath.enums.MatchType* attribute), 38
- `all()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `allFalse()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `allTrue()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `and_` (*fhirpath.enums.OPERATOR* attribute), 38
- `and_()` (in module *fhirpath.fql*), 36
- `and_()` (in module *fhirpath.fql.expressions*), 32
- `ANY` (*fhirpath.enums.MatchType* attribute), 38
- `anyFalse()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `anyTrue()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `ap()` (*fhirpath.enums.OPERATOR* method), 39
- `ap()` (in module *fhirpath.enums*), 40
- `append()` (*fhirpath.engine.base.EngineResultBody* method), 27
- `append()` (*fhirpath.engine.EngineResultBody* method), 30
- `apply_base_resource_params()`
(*fhirpath.fhirs.spec.FHIRSearchSpec* method), 30
- `apply_from_constraint()`
(*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `apply_limit()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `apply_nested()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `apply_path_replacement()`
(*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `apply_sort()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `apply_source_filter()`
(*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `as_()` (*fhirpath.fhirpath.FHIRPath* method), 41
- `ASC` (*fhirpath.enums.SortOrderType* attribute), 39
- `async_result` (*fhirpath.search.SearchContext* attribute), 59
- `AsyncElasticsearchEngine` (class in *fhirpath.engine.es*), 28
- `AsyncQueryResult` (class in *fhirpath.query*), 55
- `AsyncSearch` (class in *fhirpath.search*), 57
- `attach_callback()` (*fhirpath.thirdparty.peewee.Proxy* method), 36
- `attach_elements_terms()` (*fhirpath.search.Search* method), 57
- `attach_entry()` (*fhirpath.utils.BundleWrapper* method), 63
- `attach_limit_terms()` (*fhirpath.search.Search* method), 57
- `attach_links()` (*fhirpath.utils.BundleWrapper* method), 63
- `attach_nested_on_demand()`
(*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `attach_sort_terms()` (*fhirpath.search.Search* method), 57
- `attach_summary_terms()` (*fhirpath.search.Search* method), 57
- `attach_where_terms()` (*fhirpath.search.Search* method), 57
- `augment_with_types()`
(*fhirpath.search.SearchContext* method), 59

B

- `BaseTerm` (class in *fhirpath.fql.types*), 33
- `baseType` (*fhirpath.fhirpath.ClassInfo* attribute), 40
- `baseType` (*fhirpath.fhirpath.SimpleTypeInfo* attribute), 54
- `before_execute()` (*fhirpath.engine.base.Engine* method), 27
- `before_execute()` (*fhirpath.engine.Engine* method), 29

- `bind()` (*fhirpath.dialects.base.DialectBase* method), 25
- `bind()` (*fhirpath.query.QueryBuilder* method), 56
- `body` (*fhirpath.engine.base.EngineResult* attribute), 27
- `body` (*fhirpath.engine.EngineResult* attribute), 29
- `build()` (*fhirpath.search.Search* method), 57
- `build_elements()` (*fhirpath.fhirpath.ClassInfo* static method), 40
- `build_elements()` (*fhirpath.fhirpath.TupleTypeInfo* static method), 54
- `build_fhir_abstract_type_info()`
(*fhirpath.fhirpath.FHIRPath* static method), 41
- `build_security_query()`
(*fhirpath.engine.es.ElasticsearchEngineBase* method), 28
- `build_simple_type_info()`
(*fhirpath.fhirpath.ClassInfoElement* method), 40
- `build_type_info_from_el()`
(*fhirpath.fhirpath.FHIRPath* static method), 42
- `builder()` (in module *fhirpath.utils*), 64
- `BundleWrapper` (class in *fhirpath.utils*), 63
- C**
- `calculate_fhir_base_url()`
(*fhirpath.utils.BundleWrapper* static method), 63
- `calculate_field_index_name()`
(*fhirpath.engine.es.ElasticsearchEngineBase* method), 28
- `ceiling()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `children` (*fhirpath.utils.PathInfoContext* property), 64
- `children()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `ClassInfo` (class in *fhirpath.fhirpath*), 40
- `ClassInfoElement` (class in *fhirpath.fhirpath*), 40
- `clean_up()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 25
- `clear()` (*fhirpath.thirdparty.werkzeug.ImmutableDictMixin* method), 37
- `clone()` (*fhirpath.fhirspec.spec.SearchParameter* method), 31
- `clone()` (*fhirpath.fql.ElementPath* method), 35
- `clone()` (*fhirpath.fql.types.BaseTerm* method), 33
- `clone()` (*fhirpath.fql.types.ElementPath* method), 33
- `clone()` (*fhirpath.fql.types.ExistsGroupTerm* method), 33
- `clone()` (*fhirpath.fql.types.ExistsTerm* method), 34
- `clone()` (*fhirpath.fql.types.GroupTerm* method), 34
- `clone()` (*fhirpath.fql.types.TermValue* method), 35
- `clone()` (*fhirpath.query.Query* method), 55
- `clone()` (*fhirpath.query.QueryBuilder* method), 56
- `code` (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31
- `code` (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31
- `collection_type_required()` (in module *fhirpath.fhirpath*), 55
- `combine()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `comparator` (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31
- `comparator` (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31
- `compile()` (*fhirpath.dialects.base.DialectBase* method), 25
- `compile()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* method), 25
- `compile_for_single_resource_type()`
(*fhirpath.dialects.elasticsearch.ElasticSearchDialect* method), 25
- `component` (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31
- `component` (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31
- `concat` (*fhirpath.enums.OPERATOR* attribute), 39
- `Connection` (class in *fhirpath.connectors.connection*), 23
- `Connection` (class in *fhirpath.engine*), 29
- `ConnectionFactory` (class in *fhirpath.connectors.factory*), 23
- `ConstraintNotSatisfied`, 40
- `contained()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `contains` (*fhirpath.enums.OPERATOR* attribute), 39
- `contains()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `contains_()` (in module *fhirpath.fql*), 36
- `context_from_path()` (*fhirpath.utils.PathInfoContext* class method), 64
- `convert_and_cache_elements()`
(*fhirpath.fhirpath.FHIRPath* static method), 42
- `convertsToBoolean()` (*fhirpath.fhirpath.FHIRPath* method), 42
- `convertsToDate()` (*fhirpath.fhirpath.FHIRPath* method), 43
- `convertsToDateTime()` (*fhirpath.fhirpath.FHIRPath* method), 43
- `convertsToDecimal()` (*fhirpath.fhirpath.FHIRPath* method), 43
- `convertsToInteger()` (*fhirpath.fhirpath.FHIRPath* method), 43
- `convertsToQuantity()` (*fhirpath.fhirpath.FHIRPath* method), 44
- `convertsToString()` (*fhirpath.fhirpath.FHIRPath* method), 44
- `convertsToTime()` (*fhirpath.fhirpath.FHIRPath* method), 44
- `copy()` (*fhirpath.thirdparty.werkzeug.ImmutableDict* method), 37
- `COUNT` (*fhirpath.enums.EngineQueryType* attribute), 38
- `count()` (*fhirpath.fhirpath.FHIRPath* method), 44
- `count()` (*fhirpath.query.AsyncQueryResult* method), 55

- [count\(\)](#) (*fhirpath.query.QueryResult* method), 56
[count_raw\(\)](#) (*fhirpath.query.QueryResult* method), 56
[COUPLED](#) (*fhirpath.enums.GroupType* attribute), 38
[create\(\)](#) (*fhirpath.model.Model* static method), 55
[create_address_term\(\)](#) (*fhirpath.search.Search* method), 57
[create_codeableconcept_term\(\)](#) (*fhirpath.search.Search* method), 57
[create_coding_term\(\)](#) (*fhirpath.search.Search* method), 57
[create_connection\(\)](#) (*fhirpath.engine.base.Engine* method), 27
[create_connection\(\)](#) (*fhirpath.engine.Engine* method), 29
[create_connection\(\)](#) (in module *fhirpath.connectors*), 25
[create_contactpoint_term\(\)](#) (*fhirpath.search.Search* method), 57
[create_contains_term\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_dialect\(\)](#) (*fhirpath.engine.base.Engine* method), 27
[create_dialect\(\)](#) (*fhirpath.engine.Engine* method), 29
[create_dotted_path\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_eb_term\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_exists_term\(\)](#) (*fhirpath.search.Search* method), 57
[create_humanname_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_identifier_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_money_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_period_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_quantity_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_reference_term\(\)](#) (*fhirpath.search.Search* method), 58
[create_sa_term\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_structure\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_term\(\)](#) (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
[create_term\(\)](#) (*fhirpath.fql.types.InTerm* method), 34
[create_term\(\)](#) (*fhirpath.search.Search* method), 58
[current_url\(\)](#) (*fhirpath.engine.es.ElasticsearchEngineBase* method), 28
- ## D
- [DDL](#) (*fhirpath.enums.EngineQueryType* attribute), 38
[DECOUPLED](#) (*fhirpath.enums.GroupType* attribute), 38
[DEFAULT](#) (*fhirpath.enums.FHIR_VERSION* attribute), 38
[definitions](#) (*fhirpath.search.SearchContext* attribute), 59
[delete\(\)](#) (*fhirpath.storage.MemoryStorage* method), 59
[DESC](#) (*fhirpath.enums.SortOrderType* attribute), 39
[descendants\(\)](#) (*fhirpath.fhirpath.FHIRPath* method), 44
[DialectBase](#) (class in *fhirpath.dialects.base*), 25
[distinct\(\)](#) (*fhirpath.fhirpath.FHIRPath* method), 45
[DML](#) (*fhirpath.enums.EngineQueryType* attribute), 38
[download_and_extract\(\)](#) (in module *fhirpath.fhirspec.downloader*), 30
[download_archive\(\)](#) (in module *fhirpath.fhirspec.downloader*), 30
[DSTU2](#) (*fhirpath.enums.FHIR_VERSION* attribute), 38
- ## E
- [eb\(\)](#) (*fhirpath.enums.OPERATOR* method), 39
[eb\(\)](#) (in module *fhirpath.enums*), 40
[eb_\(\)](#) (in module *fhirpath.fql*), 36
[ElasticSearchDialect](#) (class in *fhirpath.dialects.elasticsearch*), 25
[ElasticsearchEngine](#) (class in *fhirpath.engine.es*), 28
[ElasticsearchEngineBase](#) (class in *fhirpath.engine.es*), 28
[element](#) (*fhirpath.fhirpath.ClassInfo* attribute), 40
[element](#) (*fhirpath.fhirpath.TupleTypeInfo* attribute), 54
[element\(\)](#) (*fhirpath.query.QueryBuilder* method), 56
[element_from_predecessor\(\)](#) (*fhirpath.fhirpath.FHIRPath* static method), 45
[ElementClause](#) (class in *fhirpath.fql.types*), 33
[ElementPath](#) (class in *fhirpath.fql.types*), 35
[ElementPath](#) (class in *fhirpath.fql.types*), 33
[elements](#) (*fhirpath.engine.base.EngineResultHeader* attribute), 27
[elements](#) (*fhirpath.engine.EngineResultHeader* attribute), 30
[elementType](#) (*fhirpath.fhirpath.ListTypeInfo* attribute), 54
[empty](#) (*fhirpath.fql.types.FqlClause* property), 34
[empty](#) (*fhirpath.fql.types.LimitClause* property), 34
[empty\(\)](#) (*fhirpath.fhirpath.FHIRPath* method), 45
[empty\(\)](#) (*fhirpath.query.AsyncQueryResult* method), 55
[empty\(\)](#) (*fhirpath.query.QueryResult* method), 56
[empty\(\)](#) (*fhirpath.storage.MemoryStorage* method), 59
[EmptyPathInfoContext](#) (class in *fhirpath.utils*), 64
[endsWith\(\)](#) (*fhirpath.fhirpath.FHIRPath* method), 45
[ENDWITH](#) (*fhirpath.enums.TermMatchType* attribute), 39
[Engine](#) (class in *fhirpath.engine*), 29
[Engine](#) (class in *fhirpath.engine.base*), 27
[engine](#) (*fhirpath.search.SearchContext* attribute), 59

EngineQueryType (class in *fhirpath.enums*), 38
EngineResult (class in *fhirpath.engine*), 29
EngineResult (class in *fhirpath.engine.base*), 27
EngineResultBody (class in *fhirpath.engine*), 29
EngineResultBody (class in *fhirpath.engine.base*), 27
EngineResultHeader (class in *fhirpath.engine*), 30
EngineResultHeader (class in *fhirpath.engine.base*), 27
EngineResultRow (class in *fhirpath.engine*), 30
EngineResultRow (class in *fhirpath.engine.base*), 27
ensure_spec_jsons() (in module *fhirpath.fhirspec*), 32
ensure_term_value() (*fhirpath.fql.types.BaseTerm* method), 33
ensure_term_value() (*fhirpath.fql.types.NonFhirTerm* method), 34
ensure_term_value() (*fhirpath.fql.types.Term* method), 35
eq (*fhirpath.enums.OPERATOR* attribute), 39
escape_all() (in module *fhirpath.dialects.elasticsearch*), 26
escape_star() (in module *fhirpath.dialects.elasticsearch*), 26
EXACT (*fhirpath.enums.TermMatchType* attribute), 39
exact_() (in module *fhirpath.fql*), 36
exclude() (*fhirpath.fhirpath.FHIRPath* method), 45
execute() (*fhirpath.engine.es.AsyncElasticsearchEngine* method), 28
execute() (*fhirpath.engine.es.ElasticsearchEngine* method), 28
exists() (*fhirpath.fhirpath.FHIRPath* method), 45
exists() (*fhirpath.storage.MemoryStorage* method), 59
exists_() (in module *fhirpath.fql*), 36
exists_() (in module *fhirpath.fql.expressions*), 32
exists_group_() (in module *fhirpath.fql.expressions*), 32
ExistsGroupTerm (class in *fhirpath.fql.types*), 33
ExistsTerm (class in *fhirpath.fql.types*), 34
exp() (*fhirpath.fhirpath.FHIRPath* method), 45
expand_path() (in module *fhirpath.utils*), 64
expression (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31
expression_map (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31
extract_hits() (*fhirpath.engine.es.ElasticsearchEngineBase* method), 28
extract_ids() (*fhirpath.engine.base.EngineResult* method), 27
extract_ids() (*fhirpath.engine.EngineResult* method), 29
extract_references() (*fhirpath.engine.base.EngineResult* method), 27
extract_references() (*fhirpath.engine.EngineResult* method), 29
extract_spec_files() (in module *fhirpath.fhirspec.downloader*), 30

F

fallback_callable() (in module *fhirpath.utils*), 64
fetchall() (*fhirpath.query.AsyncQueryResult* method), 55
fetchall() (*fhirpath.query.QueryResult* method), 57
FHIR_REST_SERVER_PATH_PATTERN (*fhirpath.utils.BundleWrapper* attribute), 63
fhir_rest_server_path_pattern() (*fhirpath.utils.BundleWrapper* class method), 63
fhir_search() (in module *fhirpath.search*), 59
FHIR_VERSION (class in *fhirpath.enums*), 38
FhirBase64Binary (class in *fhirpath.types*), 60
FhirbaseEngine (class in *fhirpath.engine.fhirbase*), 29
FhirBoolean (class in *fhirpath.types*), 60
FhirCanonical (class in *fhirpath.types*), 60
FhirCode (class in *fhirpath.types*), 60
FhirDate (class in *fhirpath.types*), 60
FhirDateTime (class in *fhirpath.types*), 60
FhirDecimal (class in *fhirpath.types*), 61
FhirId (class in *fhirpath.types*), 61
FhirInstant (class in *fhirpath.types*), 61
FhirInteger (class in *fhirpath.types*), 61
FhirMarkdown (class in *fhirpath.types*), 62
FhirOid (class in *fhirpath.types*), 62
fhirpath module, 65
FHIRPath (class in *fhirpath.fhirpath*), 41
fhirpath.connectors module, 25
fhirpath.connectors.connection module, 23
fhirpath.connectors.factory module, 23
fhirpath.connectors.interfaces module, 24
fhirpath.connectors.url module, 24
fhirpath.constraints module, 38
fhirpath.dialects module, 27
fhirpath.dialects.base module, 25
fhirpath.dialects.elasticsearch module, 25
fhirpath.dialects.postgres module, 26

fhirpath.dialects.sqlalchemy
 module, 26
 fhirpath.engine
 module, 29
 fhirpath.engine.base
 module, 27
 fhirpath.engine.es
 module, 28
 fhirpath.engine.fhirbase
 module, 29
 fhirpath.enums
 module, 38
 fhirpath.exceptions
 module, 40
 fhirpath.fhirpath
 module, 40
 fhirpath.fhirspec
 module, 32
 fhirpath.fhirspec.downloader
 module, 30
 fhirpath.fhirspec.spec
 module, 30
 fhirpath.fql
 module, 35
 fhirpath.fql.expressions
 module, 32
 fhirpath.fql.types
 module, 33
 fhirpath.interfaces
 module, 36
 fhirpath.interfaces.base
 module, 36
 fhirpath.interfaces.connectors
 module, 36
 fhirpath.interfaces.dialects
 module, 36
 fhirpath.interfaces.engine
 module, 36
 fhirpath.interfaces.fql
 module, 36
 fhirpath.model
 module, 55
 fhirpath.query
 module, 55
 fhirpath.search
 module, 57
 fhirpath.storage
 module, 59
 fhirpath.thirdparty
 module, 37
 fhirpath.thirdparty.peewee
 module, 36
 fhirpath.thirdparty.werkzeug
 module, 37
 fhirpath.types
 module, 60
 fhirpath.utils
 module, 63
 fhirpath.version
 module, 65
 FhirPositiveInt (class in fhirpath.types), 62
 FHIRSearchSpec (class in fhirpath.fhirspec.spec), 30
 FHIRSearchSpecFactory (class in fhirpath.fhirspec), 32
 FhirSpecFactory (class in fhirpath.fhirspec), 32
 FhirString (class in fhirpath.types), 62
 FhirTime (class in fhirpath.types), 62
 FhirUnsignedInt (class in fhirpath.types), 63
 FhirURI (class in fhirpath.types), 63
 FhirURL (class in fhirpath.types), 63
 FhirUUID (class in fhirpath.types), 63
 finalize() (fhirpath.fql.ElementPath method), 35
 finalize() (fhirpath.fql.types.BaseTerm method), 33
 finalize() (fhirpath.fql.types.ElementPath method), 33
 finalize() (fhirpath.fql.types.ExistsGroupTerm method), 33
 finalize() (fhirpath.fql.types.ExistsTerm method), 34
 finalize() (fhirpath.fql.types.GroupTerm method), 34
 finalize() (fhirpath.fql.types.InTerm method), 34
 finalize() (fhirpath.fql.types.NonFhirTerm method), 34
 finalize() (fhirpath.fql.types.SortTerm method), 35
 finalize() (fhirpath.fql.types.Term method), 35
 finalize() (fhirpath.fql.types.TermValue method), 35
 finalize() (fhirpath.query.QueryBuilder method), 56
 first() (fhirpath.fhirpath.FHIRPath method), 45
 first() (fhirpath.query.AsyncQueryResult method), 55
 first() (fhirpath.query.QueryResult method), 57
 floor() (fhirpath.fhirpath.FHIRPath method), 46
 force_bytes() (in module fhirpath.utils), 64
 force_str() (in module fhirpath.utils), 64
 fql() (in module fhirpath.fql.expressions), 32
 FqlClause (class in fhirpath.fql.types), 34
 from_() (fhirpath.query.QueryBuilder method), 56
 from_builder() (fhirpath.query.Query class method), 55
 from_config() (fhirpath.connectors.connection.Connection class method), 23
 from_config() (fhirpath.engine.Connection class method), 29
 from_definition() (fhirpath.fhirspec.spec.SearchParameter class method), 31
 from_dict() (fhirpath.fhirspec.spec.SearchParameterDefinition class method), 31
 from_el_path() (fhirpath.fql.ElementPath class method), 35
 from_el_path() (fhirpath.fql.types.ElementPath class method), 33

`from_expression()` (*fhirpath.fql.types.PathWhereConstraint* class method), 34

`from_model()` (*fhirpath.fhirpath.ClassInfo* class method), 40

`from_model()` (*fhirpath.fhirpath.TupleTypeInfo* class method), 54

`from_model_field()` (*fhirpath.fhirpath.ClassInfoElement* class method), 41

`from_params()` (*fhirpath.search.Search* class method), 58

`from_prepared()` (*fhirpath.connectors.connection.Connection* class method), 23

`from_prepared()` (*fhirpath.engine.Connection* class method), 29

`from_query_string()` (*fhirpath.search.Search* class method), 58

`from_release()` (*fhirpath.fhirspec.FHIRSearchSpecFactory* static method), 32

`from_release()` (*fhirpath.fhirspec.FhirSpecFactory* static method), 32

`from_specifier()` (*fhirpath.fhirpath.ListTypeInfo* class method), 54

`from_type_specifier()` (*fhirpath.fhirpath.SimpleTypeInfo* class method), 54

`from_url()` (*fhirpath.connectors.connection.Connection* class method), 23

`from_url()` (*fhirpath.engine.Connection* class method), 29

`FromClause` (class in *fhirpath.fql.types*), 34

`fromkeys()` (*fhirpath.thirdparty.werkzeug.ImmutableDictMixin* class method), 37

`FULLTEXT` (*fhirpath.enums.TermMatchType* attribute), 39

`get_elements()` (*fhirpath.fhirpath.ListTypeInfo* method), 54

`get_elements()` (*fhirpath.fhirpath.TupleTypeInfo* method), 54

`get_expression()` (*fhirpath.fhirspec.spec.SearchParameter* method), 31

`get_from()` (*fhirpath.query.Query* method), 56

`get_index_name()` (*fhirpath.engine.es.ElasticsearchEngineBase* method), 28

`get_limit()` (*fhirpath.query.Query* method), 56

`get_local_timezone()` (in module *fhirpath.utils*), 64

`get_mapping()` (*fhirpath.engine.es.ElasticsearchEngineBase* method), 28

`get_parameters_definition()` (*fhirpath.search.SearchContext* method), 59

`get_path_mapping_info()` (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26

`get_query()` (*fhirpath.query.QueryBuilder* method), 56

`get_real_value()` (*fhirpath.fql.types.BaseTerm* method), 33

`get_real_value()` (*fhirpath.fql.types.NonFhirTerm* method), 34

`get_real_value()` (*fhirpath.fql.types.Term* method), 35

`get_select()` (*fhirpath.query.Query* method), 56

`get_sort()` (*fhirpath.query.Query* method), 56

`get_type()` (*fhirpath.fhirpath.FHIRPath* method), 46

`get_version()` (in module *fhirpath*), 65

`get_where()` (*fhirpath.query.Query* method), 56

`GroupTerm` (class in *fhirpath.fql.types*), 34

`GroupType` (class in *fhirpath.enums*), 38

`gt` (*fhirpath.enums.OPERATOR* attribute), 39

G

`G_()` (in module *fhirpath.fql*), 35

`G_()` (in module *fhirpath.fql.expressions*), 32

`ge` (*fhirpath.enums.OPERATOR* attribute), 39

`generate_mappings()` (*fhirpath.engine.es.ElasticsearchEngineBase* method), 28

`generated_on` (*fhirpath.engine.base.EngineResultHeader* attribute), 27

`generated_on` (*fhirpath.engine.EngineResultHeader* attribute), 30

`get()` (*fhirpath.storage.MemoryStorage* method), 59

`get_backend_name()` (*fhirpath.connectors.url.URL* method), 24

`get_driver_name()` (*fhirpath.connectors.url.URL* method), 24

`get_element()` (*fhirpath.query.Query* method), 56

`get_elements()` (*fhirpath.fhirpath.ClassInfo* method), 40

H

`has()` (*fhirpath.search.Search* method), 58

`has_escape_comma()` (in module *fhirpath.search*), 59

`header` (*fhirpath.engine.base.EngineResult* attribute), 27

`header` (*fhirpath.engine.EngineResult* attribute), 29

I

`iff()` (*fhirpath.fhirpath.FHIRPath* method), 46

`ImmutableDict` (class in *fhirpath.thirdparty.werkzeug*), 37

`ImmutableDictMixin` (class in *fhirpath.thirdparty.werkzeug*), 37

`import_string()` (in module *fhirpath.utils*), 64

`in_()` (*fhirpath.fhirpath.FHIRPath* method), 46

`in_()` (in module *fhirpath.fql*), 36

`in_()` (in module *fhirpath.fql.expressions*), 33

`include()` (*fhirpath.search.Search* method), 58

`indexOf()` (*fhirpath.fhirpath.FHIRPath* method), 46

`init_data()` (*fhirpath.utils.BundleWrapper* static method), 63

- `initial_bundle_data()` (*fhirpath.engine.es.ElasticsearchEngineBase method*), 28
- `initialize()` (*fhirpath.thirdparty.peewee.Proxy method*), 37
- `insert()` (*fhirpath.storage.MemoryStorage method*), 59
- `InTerm` (*class in fhirpath.fql.types*), 34
- `intersect()` (*fhirpath.fhirpath.FHIRPath method*), 46
- `is_()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `is_async()` (*fhirpath.connectors.connection.Connection class method*), 23
- `is_async()` (*fhirpath.engine.base.Engine class method*), 27
- `is_async()` (*fhirpath.engine.Connection class method*), 29
- `is_async()` (*fhirpath.engine.Engine class method*), 29
- `is_async()` (*fhirpath.engine.es.AsyncElasticsearchEngine class method*), 28
- `is_fhir_primitive_type()` (*fhirpath.dialects.base.DialectBase static method*), 25
- `is_finalized()` (*fhirpath.fql.ElementPath method*), 35
- `is_finalized()` (*fhirpath.fql.types.ElementPath method*), 33
- `is_finalized()` (*fhirpath.fql.types.TermValue method*), 35
- `is_immutable()` (*in module fhirpath.thirdparty.werkzeug*), 37
- `is_root()` (*fhirpath.utils.PathInfoContext method*), 64
- `isDistinct()` (*fhirpath.fhirpath.FHIRPath method*), 46
- `isOneBased` (*fhirpath.fhirpath.ClassInfoElement attribute*), 41
- `isOneBased` (*fhirpath.fhirpath.TupleTypeInfoElement attribute*), 55
- `iteritems()` (*in module fhirpath.thirdparty.werkzeug*), 37
- ## J
- `json()` (*fhirpath.utils.BundleWrapper method*), 63
- `jsonfilename` (*fhirpath.fhirspec.spec.FHIRSearchSpec property*), 30
- ## L
- `last()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `last()` (*fhirpath.query.QueryResult method*), 57
- `le` (*fhirpath.enums.OPERATOR attribute*), 39
- `length()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `limit` (*fhirpath.fql.types.LimitClause property*), 34
- `limit()` (*fhirpath.query.QueryBuilder method*), 56
- `LimitClause` (*class in fhirpath.fql.types*), 34
- `ListTypeInfo` (*class in fhirpath.fhirpath*), 54
- `ln()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `log()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `lookup_all_fhir_domain_resource_classes()` (*in module fhirpath.utils*), 64
- `lookup_fhir_class()` (*in module fhirpath.utils*), 64
- `lookup_fhir_class_path()` (*in module fhirpath.utils*), 65
- `lookup_fhir_resource_spec()` (*in module fhirpath.fhirspec*), 32
- `lower()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `lt` (*fhirpath.enums.OPERATOR attribute*), 39
- ## M
- `make_link()` (*fhirpath.utils.BundleWrapper method*), 64
- `make_url()` (*in module fhirpath.connectors*), 25
- `match_all()` (*fhirpath.fql.types.ExistsGroupTerm method*), 33
- `match_all()` (*fhirpath.fql.types.GroupTerm method*), 34
- `match_any()` (*fhirpath.fql.types.ExistsGroupTerm method*), 33
- `match_any()` (*fhirpath.fql.types.GroupTerm method*), 34
- `match_no_one()` (*fhirpath.fql.types.ExistsGroupTerm method*), 33
- `match_no_one()` (*fhirpath.fql.types.GroupTerm method*), 34
- `match_one()` (*fhirpath.fql.types.ExistsGroupTerm method*), 34
- `match_one()` (*fhirpath.fql.types.GroupTerm method*), 34
- `matches()` (*fhirpath.fhirpath.FHIRPath method*), 47
- `MatchType` (*class in fhirpath.enums*), 38
- `MemoryStorage` (*class in fhirpath.storage*), 59
- `Model` (*class in fhirpath.model*), 55
- `ModelFactory` (*class in fhirpath.model*), 55
- `modifier` (*fhirpath.fhirspec.spec.SearchParameter attribute*), 31
- `modifier` (*fhirpath.fhirspec.spec.SearchParameterDefinition attribute*), 31
- `module`
- `fhirpath`, 65
 - `fhirpath.connectors`, 25
 - `fhirpath.connectors.connection`, 23
 - `fhirpath.connectors.factory`, 23
 - `fhirpath.connectors.interfaces`, 24
 - `fhirpath.connectors.url`, 24
 - `fhirpath.constraints`, 38
 - `fhirpath.dialects`, 27
 - `fhirpath.dialects.base`, 25
 - `fhirpath.dialects.elasticsearch`, 25
 - `fhirpath.dialects.postgres`, 26
 - `fhirpath.dialects.sqlalchemy`, 26
 - `fhirpath.engine`, 29
 - `fhirpath.engine.base`, 27
 - `fhirpath.engine.es`, 28
 - `fhirpath.engine.fhirbase`, 29
 - `fhirpath.enums`, 38
 - `fhirpath.exceptions`, 40

`fhirpath.fhirpath`, 40
`fhirpath.fhirspec`, 32
`fhirpath.fhirspec.downloader`, 30
`fhirpath.fhirspec.spec`, 30
`fhirpath.fql`, 35
`fhirpath.fql.expressions`, 32
`fhirpath.fql.types`, 33
`fhirpath.interfaces`, 36
`fhirpath.interfaces.base`, 36
`fhirpath.interfaces.connectors`, 36
`fhirpath.interfaces.dialects`, 36
`fhirpath.interfaces.engine`, 36
`fhirpath.interfaces.fql`, 36
`fhirpath.model`, 55
`fhirpath.query`, 55
`fhirpath.search`, 57
`fhirpath.storage`, 59
`fhirpath.thirdparty`, 37
`fhirpath.thirdparty.peewee`, 36
`fhirpath.thirdparty.werkzeug`, 37
`fhirpath.types`, 60
`fhirpath.utils`, 63
`fhirpath.version`, 65

`multiple_and` (`fhirpath.fhirspec.spec.SearchParameter` attribute), 31
`multiple_and` (`fhirpath.fhirspec.spec.SearchParameterDefinition` attribute), 31
`multiple_or` (`fhirpath.fhirspec.spec.SearchParameter` attribute), 31
`multiple_or` (`fhirpath.fhirspec.spec.SearchParameterDefinition` attribute), 31
`MultipleResultsFound`, 40

N

`name` (`fhirpath.fhirpath.ClassInfo` attribute), 40
`name` (`fhirpath.fhirpath.ClassInfoElement` attribute), 41
`name` (`fhirpath.fhirpath.SimpleTypeInfo` attribute), 54
`name` (`fhirpath.fhirpath.TupleTypeInfoElement` attribute), 55
`name` (`fhirpath.fhirspec.spec.SearchParameter` attribute), 31
`name` (`fhirpath.fhirspec.spec.SearchParameterDefinition` attribute), 31
`namespace` (`fhirpath.fhirpath.ClassInfo` attribute), 40
`namespace` (`fhirpath.fhirpath.SimpleTypeInfo` attribute), 54
`navigate_indexed_path` (in module `fhirpath.engine.es`), 29
`ne` (`fhirpath.enums.OPERATOR` attribute), 39
`neg` (`fhirpath.enums.OPERATOR` attribute), 39
`non_fhir` (`fhirpath.fql.ElementPath` property), 35
`non_fhir` (`fhirpath.fql.types.ElementPath` property), 33
`NONE` (`fhirpath.enums.MatchType` attribute), 38
`NonFhirTerm` (class in `fhirpath.fql.types`), 34
`NoResultFound`, 40
`normalize`() (`fhirpath.enums.FHIR_VERSION` static method), 38
`normalize_param`() (`fhirpath.search.SearchContext` method), 59
`normalize_param_value`() (`fhirpath.search.SearchContext` method), 59
`not_` (`fhirpath.enums.OPERATOR` attribute), 39
`not_`() (in module `fhirpath.fql`), 36
`not_`() (in module `fhirpath.fql.expressions`), 33
`not_exists_`() (in module `fhirpath.fql`), 36
`not_exists_`() (in module `fhirpath.fql.expressions`), 33
`not_in_`() (in module `fhirpath.fql`), 36
`now`() (`fhirpath.fhirpath.FHIRPath` method), 47

O

`obj` (`fhirpath.thirdparty.peewee.Proxy` attribute), 37
`obj` (`fhirpath.utils.PathInfoContextProxy` attribute), 64
`OFF__getitem__`() (`fhirpath.query.QueryResult` method), 56
`offset` (`fhirpath.fql.types.LimitClause` property), 34
`ofType`() (`fhirpath.fhirpath.FHIRPath` method), 48
`ONE` (`fhirpath.enums.MatchType` attribute), 38
`OPERATOR` (class in `fhirpath.enums`), 38
`operator` (`fhirpath.enums.OPERATOR` attribute), 39
`or_`() (in module `fhirpath.fql`), 36
`or_`() (in module `fhirpath.fql.expressions`), 33
`order` (`fhirpath.fql.types.SortTerm` attribute), 35

P

`parent` (`fhirpath.utils.PathInfoContext` property), 64
`parse`() (`fhirpath.fql.ElementPath` method), 35
`parse`() (`fhirpath.fql.types.ElementPath` method), 33
`parse_composite_parameter_component`() (`fhirpath.search.SearchContext` method), 59
`parse_query_string`() (`fhirpath.search.Search` static method), 58
`passthrough`() (`fhirpath.thirdparty.peewee.Proxy` method), 37
`password` (`fhirpath.connectors.url.URL` property), 24
`path` (`fhirpath.fql.ElementPath` property), 35
`path` (`fhirpath.fql.types.ElementPath` property), 33
`path` (`fhirpath.fql.types.SortTerm` attribute), 35
`PathInfoContext` (class in `fhirpath.utils`), 64
`PathInfoContextProxy` (class in `fhirpath.utils`), 64
`PathWhereConstraint` (class in `fhirpath.fql.types`), 34
`pop`() (`fhirpath.thirdparty.werkzeug.ImmutableDictMixin` method), 37
`popitem`() (`fhirpath.thirdparty.werkzeug.ImmutableDictMixin` method), 37
`pos` (`fhirpath.enums.OPERATOR` attribute), 39

- PostgresDialect (class in *fhirpath.dialects.postgres*), 26
- power() (*fhirpath.fhirpath.FHIRPath* method), 48
- pre_compile() (*fhirpath.dialects.base.DialectBase* method), 25
- prepare() (*fhirpath.fhirspec.spec.FHIRSearchSpec* method), 30
- prepare_params() (*fhirpath.search.Search* method), 58
- process_raw_result() (*fhirpath.engine.es.AsyncElasticsearchEngine* method), 28
- process_raw_result() (*fhirpath.engine.es.ElasticsearchEngine* method), 28
- Proxy (class in *fhirpath.thirdparty.peewee*), 36
- proxy() (in module *fhirpath.utils*), 65
- ## Q
- Q_() (in module *fhirpath.query*), 55
- Query (class in *fhirpath.query*), 55
- QueryBuilder (class in *fhirpath.query*), 56
- QueryResult (class in *fhirpath.query*), 56
- ## R
- R4 (*fhirpath.enums.FHIR_VERSION* attribute), 38
- raw_connection (*fhirpath.connectors.connection.Connection* property), 24
- raw_connection (*fhirpath.engine.Connection* property), 29
- raw_query (*fhirpath.engine.base.EngineResultHeader* attribute), 27
- raw_query (*fhirpath.engine.EngineResultHeader* attribute), 30
- repeat() (*fhirpath.fhirpath.FHIRPath* method), 48
- replace() (*fhirpath.fhirpath.FHIRPath* method), 48
- replaceMatches() (*fhirpath.fhirpath.FHIRPath* method), 48
- required_finalized() (in module *fhirpath.constraints*), 38
- required_from_resource() (in module *fhirpath.constraints*), 38
- required_not_finalized() (in module *fhirpath.constraints*), 38
- required_value_not_assigned() (in module *fhirpath.constraints*), 38
- reraise() (in module *fhirpath.utils*), 65
- resolve_absolute_uri() (*fhirpath.utils.BundleWrapper* method), 64
- resolve_datetime_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* method), 26
- resolve_exists_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
- resolve_nonfhir_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* method), 26
- resolve_numeric_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
- resolve_path_context() (*fhirpath.search.SearchContext* method), 59
- resolve_string_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* static method), 26
- resolve_term() (*fhirpath.dialects.elasticsearch.ElasticSearchDialect* method), 26
- resource_type (*fhirpath.fhirspec.spec.ResourceSearchParameterDefinition* attribute), 30
- resource_types (*fhirpath.search.SearchContext* attribute), 59
- ResourceSearchParameterDefinition (class in *fhirpath.fhirspec.spec*), 30
- response() (*fhirpath.search.Search* method), 58
- rev_include() (*fhirpath.search.Search* method), 58
- round() (*fhirpath.fhirpath.FHIRPath* method), 49
- ## S
- sa() (*fhirpath.enums.OPERATOR* method), 39
- sa() (in module *fhirpath.enums*), 40
- sa_() (in module *fhirpath.fql*), 36
- Search (class in *fhirpath.search*), 57
- search_params_intersection (*fhirpath.search.SearchContext* attribute), 59
- SearchContext (class in *fhirpath.search*), 59
- SearchParameter (class in *fhirpath.fhirspec.spec*), 30
- SearchParameterDefinition (class in *fhirpath.fhirspec.spec*), 31
- select() (*fhirpath.fhirpath.FHIRPath* method), 49
- select() (*fhirpath.query.QueryBuilder* method), 56
- SelectClause (class in *fhirpath.fql.types*), 34
- set_match_type() (*fhirpath.fql.types.BaseTerm* method), 33
- setdefault() (*fhirpath.thirdparty.werkzeug.ImmutableDictMixin* method), 37
- SimpleTypeInfo (class in *fhirpath.fhirpath*), 54
- single() (*fhirpath.fhirpath.FHIRPath* method), 49
- single() (*fhirpath.query.AsyncQueryResult* method), 55
- single() (*fhirpath.query.QueryResult* method), 57
- single_valued_address_term() (*fhirpath.search.Search* method), 58
- single_valued_codeableconcept_term() (*fhirpath.search.Search* method), 58
- single_valued_coding_term() (*fhirpath.search.Search* method), 58

`single_valued_contactpoint_term()`
(*fhirpath.search.Search method*), 58

`single_valued_humanname_term()`
(*fhirpath.search.Search method*), 58

`single_valued_identifier_term()`
(*fhirpath.search.Search method*), 58

`single_valued_money_term()`
(*fhirpath.search.Search method*), 58

`single_valued_period_term()`
(*fhirpath.search.Search method*), 58

`single_valued_quantity_term()`
(*fhirpath.search.Search method*), 58

`single_valued_reference_term()`
(*fhirpath.search.Search method*), 58

`skip()` (*fhirpath.fhirpath.FHIRPath method*), 50

`skip()` (*fhirpath.query.QueryResult method*), 57

`sort()` (*fhirpath.query.QueryBuilder method*), 56

`sort_()` (in module *fhirpath.fql*), 36

`sort_()` (in module *fhirpath.fql.expressions*), 33

`SortClause` (class in *fhirpath.fql.types*), 35

`SortOrderType` (class in *fhirpath.enums*), 39

`SortTerm` (class in *fhirpath.fql.types*), 35

`spec` (*fhirpath.fhirs.spec.spec.SearchParameterDefinition attribute*), 31

`SqlAlchemyDialect` (class in *fhirpath.dialects.sqlalchemy*), 26

`sqrt()` (*fhirpath.fhirpath.FHIRPath method*), 50

`star` (*fhirpath.fql.ElementPath property*), 35

`star` (*fhirpath.fql.types.ElementPath property*), 33

`startsWith()` (*fhirpath.fhirpath.FHIRPath method*), 50

`STARTWITH` (*fhirpath.enums.TermMatchType attribute*), 39

`STU3` (*fhirpath.enums.FHIR_VERSION attribute*), 38

`sub` (*fhirpath.enums.OPERATOR attribute*), 39

`subsetOf()` (*fhirpath.fhirpath.FHIRPath method*), 50

`substring()` (*fhirpath.fhirpath.FHIRPath method*), 50

`supersetOf()` (*fhirpath.fhirpath.FHIRPath method*), 50

`support_prefix()` (*fhirpath.fhirs.spec.spec.SearchParameterDefinition method*), 31

T

`T1` (*fhirpath.enums.WhereConstraintType attribute*), 39

`T2` (*fhirpath.enums.WhereConstraintType attribute*), 39

`T3` (*fhirpath.enums.WhereConstraintType attribute*), 40

`T_()` (in module *fhirpath.fql*), 36

`T_()` (in module *fhirpath.fql.expressions*), 32

`tail()` (*fhirpath.fhirpath.FHIRPath method*), 51

`tail()` (*fhirpath.query.QueryResult method*), 57

`take()` (*fhirpath.fhirpath.FHIRPath method*), 51

`take()` (*fhirpath.query.QueryResult method*), 57

`target` (*fhirpath.fhirs.spec.spec.SearchParameter attribute*), 31

`target` (*fhirpath.fhirs.spec.spec.SearchParameterDefinition attribute*), 31

`Term` (class in *fhirpath.fql.types*), 35

`TermMatchType` (class in *fhirpath.enums*), 39

`TermValue` (class in *fhirpath.fql.types*), 35

`timeOfDay()` (*fhirpath.fhirpath.FHIRPath method*), 51

`timestamp_local()` (in module *fhirpath.utils*), 65

`timestamp_utc()` (in module *fhirpath.utils*), 65

`to_list()` (in module *fhirpath.connectors.url*), 24

`to_python()` (*fhirpath.types.FhirBase64Binary method*), 60

`to_python()` (*fhirpath.types.FhirBoolean method*), 60

`to_python()` (*fhirpath.types.FhirCanonical method*), 60

`to_python()` (*fhirpath.types.FhirCode method*), 60

`to_python()` (*fhirpath.types.FhirDate method*), 60

`to_python()` (*fhirpath.types.FhirDateTime method*), 61

`to_python()` (*fhirpath.types.FhirDecimal method*), 61

`to_python()` (*fhirpath.types.FhirId method*), 61

`to_python()` (*fhirpath.types.FhirInstant method*), 61

`to_python()` (*fhirpath.types.FhirInteger method*), 62

`to_python()` (*fhirpath.types.FhirMarkdown method*), 62

`to_python()` (*fhirpath.types.FhirOid method*), 62

`to_python()` (*fhirpath.types.FhirPositiveInt method*), 62

`to_python()` (*fhirpath.types.FhirString method*), 62

`to_python()` (*fhirpath.types.FhirTime method*), 63

`to_python()` (*fhirpath.types.FhirUnsignedInt method*), 63

`to_python()` (*fhirpath.types.FhirURI method*), 63

`to_python()` (*fhirpath.types.FhirURL method*), 63

`to_python()` (*fhirpath.types.FhirUUID method*), 63

`toBoolean()` (*fhirpath.fhirpath.FHIRPath method*), 51

`toChars()` (*fhirpath.fhirpath.FHIRPath method*), 51

`toDate()` (*fhirpath.fhirpath.FHIRPath method*), 51

`toDateTime()` (*fhirpath.fhirpath.FHIRPath method*), 51

`today()` (*fhirpath.fhirpath.FHIRPath method*), 53

`toDecimal()` (*fhirpath.fhirpath.FHIRPath method*), 52

`toInteger()` (*fhirpath.fhirpath.FHIRPath method*), 52

`toQuantity()` (*fhirpath.fhirpath.FHIRPath method*), 52

`toString()` (*fhirpath.fhirpath.FHIRPath method*), 53

`total` (*fhirpath.engine.base.EngineResultHeader attribute*), 27

`total` (*fhirpath.engine.EngineResultHeader attribute*), 30

`total()` (*fhirpath.storage.MemoryStorage method*), 59

`toTime()` (*fhirpath.fhirpath.FHIRPath method*), 53

`trace()` (*fhirpath.fhirpath.FHIRPath method*), 53

`translate_connect_args()`
(*fhirpath.connectors.url.URL method*), 24

`truncate()` (*fhirpath.fhirpath.FHIRPath method*), 53

`TupleTypeInfo` (class in *fhirpath.fhirpath*), 54

`TupleTypeInfoElement` (class in *fhirpath.fhirpath*), 55

`type` (*fhirpath.fhirpath.ClassInfoElement attribute*), 41

`type` (*fhirpath.fhirpath.TupleTypeInfoElement attribute*), 55

`type` (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31
`xpath` (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31
`type` (*fhirpath.fhirspec.spec.SearchParameterDefinition* attribute), 31

U

`union()` (*fhirpath.fhirpath.FHIRPath* method), 53
`unrestricted` (*fhirpath.search.SearchContext* attribute), 59
`unwrap_proxy()` (in module *fhirpath.utils*), 65
`update()` (*fhirpath.thirdparty.werkzeug.ImmutableDictMixin* method), 37
`upper()` (*fhirpath.fhirpath.FHIRPath* method), 54
`URL` (class in *fhirpath.connectors.url*), 24

V

`V_()` (in module *fhirpath.fql*), 36
`V_()` (in module *fhirpath.fql.expressions*), 32
`validate()` (*fhirpath.fql.ElementPath* method), 35
`validate()` (*fhirpath.fql.types.BaseTerm* method), 33
`validate()` (*fhirpath.fql.types.ElementPath* method), 33
`validate()` (*fhirpath.fql.types.NonFhirTerm* method), 34
`validate()` (*fhirpath.fql.types.Term* method), 35
`validate()` (*fhirpath.search.Search* method), 58
`validate_normalized_value()`
 (*fhirpath.search.Search* static method), 58
`validate_params()` (*fhirpath.search.Search* static method), 58
`validate_pre_term()` (*fhirpath.search.Search* method), 58
`validate_value()` (*fhirpath.utils.PathInfoContext* method), 64
`ValidationError`, 40

W

`where()` (*fhirpath.fhirpath.FHIRPath* method), 54
`where()` (*fhirpath.query.QueryBuilder* method), 56
`WhereClause` (class in *fhirpath.fql.types*), 35
`WhereConstraintType` (class in *fhirpath.enums*), 39
`wrap()` (*fhirpath.connectors.factory.ConnectionFactory* method), 23
`wrapped_with_bundle()`
 (*fhirpath.engine.es.ElasticsearchEngineBase* method), 29
`write()` (*fhirpath.fhirspec.spec.FHIRSearchSpec* method), 30

X

`xor` (*fhirpath.enums.OPERATOR* attribute), 39
`xor_()` (in module *fhirpath.fql.expressions*), 33
`xpath` (*fhirpath.fhirspec.spec.SearchParameter* attribute), 31